

[illegible]

```

LL               IIIIII      SSSSSSSS
LL              IIIIIII     SSSSSSSS
LL             III         SS
LL            II          SS
LL           II          SS
LL          II          SS
LL         II          SSSSSS
LL        II          SSSSSS
LL       II          SS
LL      II          SS
LL     II          SS
LL    II          SSSSSS
LLLLLLLLLLLLLLL IIIIIII   SSSSSSSS
LLLLLLLLLLLLLLL IIIIIII   SSSSSSSS

```

```
0001 0 XTITLE 'LIB$SCREEN - Screen Management'
0002 0 MODULE LIB$SCREEN (
0003 0 IDENT = 'V04-000' ! File: SCREEN.B32 Edit: PLL1011
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: General Utility Library
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module contains routines to perform terminal graphics
0037 1 functions on the logical device SYS$OUTPUT. The terminal type
0038 1 is interrogated and saved on the first routine call and
0039 1 determines the escape sequences to be sent to the terminal.
0040 1 For unknown terminals and terminals without any graphic
0041 1 functions (LA36), an appropriate sequence is sent when
0042 1 necessary (carriage return instead of SET_CURSOR.
0043 1
0044 1 ENVIRONMENT: User mode, Shared library routines.
0045 1
0046 1 AUTHOR: R. Reichert, CREATION DATE: 25-AUG-1982
0047 1
0048 1 MODIFIED BY:
0049 1
0050 1 1-001 - Original.
0051 1 This module is based on SCRPKG.MAR, version V03-001,
0052 1 dated 13-April-1982.
0053 1 RKR 25-AUG-1982
0054 1 1-002 - Use VMS logicals to point to require files. PLL 24-Jan-1983
0055 1 1-003 - Don't check the return status from FILL_MAP, to remain
0056 1 compatible with the original macro code. PLL 31-Jan-1983
0057 1 1-004 - In FILL_MAP, accept end_index equal to scr$l_area. PLL 3-Feb-1983
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

F 15

16-Sep-1984 02:28:18

14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742

[VMSLIB.SRC]SCREEN.B32;1

Page 2
(1)

```
: 58      0058 1 1-005 - LIB$ERASE_LINE, LIB$ERASE_PAGE, and LIB$SET_SCROLL should treat
: 59      0059 1 a single null argument as no arguments passed. PLL 22-Feb-1983
: 60      0060 1 1-006 - Screwed up the last edit - try again. PLL 24-Feb-1983
: 61      0061 1 1-007 - Correct INVCHA error - it should be IVCHAN. PLL 19-May-1983
: 62      0062 1 1-008 - IVCHAN doesn't exist. Change back to INVCHA to allow system to build
: 63      0063 1 ADE 29-May-1983
: 64      0064 1 1-009 - In SCR$PUT_BUFFER, change call to SCR$PUT_SCREEN to OUTPUT. Try
: 65      0065 1 again to eliminate INVCHA error message. PLL 19-Jul-1983
: 66      0066 1 1-010 - To prevent an access violation, make sure that macro $SCR_APPEND
: 67      0067 1 doesn't try to move a negative number of bytes. PLL 9-Jul-1984
: 68      0068 1 1-011 - When buffering is turned off, make sure the bufsiz in the TCB is
: 69      0069 1 returned to the non-buffered length so subsequent non-buffered
: 70      0070 1 output isn't truncated. PLL 10-Jul-1984
: 71      0071 1 --
: 72      0072 1
```

```

74 0073 1 %SBTTL 'Declarations'
75 0074 1
76 0075 1 SWITCHES:
77 0076 1
78 0077 1
79 0078 1
80 0079 1 LINKAGES:
81 0080 1
82 0081 1 NONE
83 0082 1
84 0083 1 INCLUDE FILES:
85 0084 1
86 0085 1
87 0086 1 REQUIRE 'SRC$:SCRPROLOG';      ! defines psects, macros, &
88 0161 1                               ! terminal defs
89 0162 1 REQUIRE 'LIB$:STRLNK';        ! JSB linkages for STR$ routines
90 0347 1
91 0348 1 TABLE OF CONTENTS:
92 0349 1
93 0350 1
94 0351 1 FORWARD ROUTINE
95 0352 1
96 0353 1 ! The LIB$ entry points
97 0354 1
98 0355 1 LIB$ERASE_LINE,      ! Erase Line from Screen
99 0356 1 LIB$ERASE_PAGE,      ! Erase Page from Screen
100 0357 1 LIB$PUT_BUFFER,      ! Put Current Buffer to Screen or to Prev. Buffer
101 0358 1 LIB$PUT_LINE,        ! Put Text to Screen in Line Mode
102 0359 1 LIB$PUT_SCREEN,      ! Put Text to Screen
103 0360 1 LIB$SET_CURSOR,      ! Set Cursor to Character Position on Screen
104 0361 1 LIB$SET_SCROLL,      ! Set Scrolling Region
105 0362 1
106 0363 1 ! The SCR$ entry points
107 0364 1
108 0365 1 SCR$DOWN_SCROLL,      ! Down Scroll, Move Cursor up One Line
109 0366 1 SCR$ERASE_LINE,      ! Erase Line
110 0367 1 SCR$ERASE_PAGE,      ! Erase Page
111 0368 1 SCR$PUT_BUFFER,      ! Put Current Buffer to Screen or Previous Buffer
112 0369 1 SCR$PUT_LINE,        ! Put Text to Screen in Line Mode
113 0370 1 SCR$PUT_SCREEN,      ! Put Text to Screen
114 0371 1 SCR$SET_BUFFER,      ! Set/Clear Buffer Mode
115 0372 1 SCR$SET_CURSOR,      ! Set Cursor to Character Position on Screen
116 0373 1 SCR$SET_SCROLL,      ! Establish Scrolling Region
117 0374 1 SCR$UP_SCROLL,       ! Up Scroll, Move Cursor Down One Line
118 0375 1
119 0376 1 ! The following routine was completely commented out in the
120 0377 1 ! original Macro version, SCRPKG. Included here only for
121 0378 1 ! historical reasons.
122 0379 1 ! SCR$MOVE_CURSOR,      ! Move Cursor to Relative Location
123 0380 1
124 0381 1 ! Internally used routines
125 0382 1
126 0383 1 SCR$$GET_TYPE_R3 : GET_TYPE_LINK, ! Get device type
127 0384 1 SCR$$FOREIGN,          ! Foreign Terminal Handler
128 0385 1
129 0386 1 ! Local subroutines and functions
130 0387 1
```

```
131 0388 1 DO_ATTR,      ! Output simulated bolding and underlining.
132 0389 1 FILL_MAP,    ! Fill map with specified character
133 0390 1 OUTPUT,      ! Write string to terminal
134 0391 1 PUT_MAP,     ! Output virtual screen map
135 0392 1 RMS_OUTPUT,  ! Write a string via RMS
136 0393 1 SET_CURSOR, ! Create set cursor sequence
137 0394 1 TRIMMED_LENGTH; ! Calc. length of string without trailing blanks
138 0395 1
139 0396 1
140 0397 1 +
141 0398 1   The following routine entry point names are defined as alias for
142 0399 1   routines defined above. They are defined via
143 0400 1   GLOBAL BIND ROUTINE LIB$xxx = SCR$xxx declarations.
144 0401 1 -
145 0402 1   LIB$DOWN_SCROLL = SCR$DOWN_SCROLL ! Down scroll, move cursor up 1 line
146 0403 1   LIB$SET_BUFFER = SCR$SET_BUFFER ! Set or Clear Screen Buffer Mode
147 0404 1   LIB$UP_SCROLL  = SCR$UP_SCROLL  ! Up scroll, move cursor down 1 line
148 0405 1
149 0406 1 +
150 0407 1   The following entry point is obsolete, but is still present in
151 0408 1   SCRVECTOR. So it's defined here to keep the linker happy.
152 0409 1 -
153 0410 1   SCR$ERASE      = SCR$ERASE_PAGE
154 0411 1
155 0412 1
156 0413 1   MACROS:
157 0414 1
158 0415 1
159 0416 1
160 0417 1   EQUATED SYMBOLS:
161 0418 1
162 0419 1
163 0420 1
164 0421 1   FIELDS:
165 0422 1
166 0423 1       NONE
167 0424 1
168 0425 1   PSECTS:
169 0426 1
170 0427 1
171 0428 1   OWN STORAGE:
172 0429 1
173 0430 1 +
174 0431 1       W R I T A B L E   D A T A   D E F I N I T I O N S
175 0432 1       -----
176 0433 1 -
177 0434 1 GLOBAL
178 0435 1     SCR$L_FLINKHEAD : INITIAL (0),      ! Initial state is empty
179 0436 1     SCR$L_CUROUTPUT  : INITIAL (0);      ! Pointer to current output unit
180 0437 1
181 0438 1 OWN
182 0439 1     SCR$L_WORKMASK   : INITIAL (0);      ! Used to reset attribute mask
183 0440 1
184 0441 1
185 0442 1
186 0443 1
187 0444 1
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
Declarations

I 15
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 5
(2)

```
: 188      0445 1 |
: 189      0446 1 | EXTERNAL REFERENCES:
: 190      0447 1 |
: 191      0448 1 |
: 192      0449 1 | EXTERNAL ROUTINE
: 193      0450 1 |   LIB$ANALYZE_SDESC_R2 : LIB$ANALYZE_SDESC_JSB_LINK,
: 194      0451 1 |       Get length and address from descriptor
: 195      0452 1 |   LIB$ASSIGN,       Assign channel
: 196      0453 1 |   LIB$CALL_IMAGE,   Call image
: 197      0454 1 |   LIB$FREE_EF,      Deallocate event flag number
: 198      0455 1 |   LIB$FREE_VM,      Deallocate heap storage
: 199      0456 1 |   LIB$GET_EF,        Get local event flag
: 200      0457 1 |   LIB$GET_INPUT,     Get input from SYSSINPUT
: 201      0458 1 |   LIB$GET_VM,        Allocate heap storage
: 202      0459 1 |   LIB$LP_LINES,      Get default lines per page
: 203      0460 1 |   LIB$PUT_OUTPUT,    Output to SYSSOUTPUT
: 204      0461 1 |   LIB$SFREE1_DD,     Release space for a dynamic string
: 205      0462 1 |   LIB$SIG_TO_RET,    Convert signals to return status
: 206      0463 1 |   SCR$SET_OUTPUT,    Establish terminal for output
: 207      0464 1 |   SCR$STOP_OUTPUT,   Stop output to terminal or screen buffer
: 208      0465 1 |   STR$CONCAT,        Concatenate several strings into one
: 209      0466 1 |   STR$DUPL_CHAR;     Fill a string with some character
: 210      0467 1 |
: 211      0468 1 | EXTERNAL LITERAL
: 212      0469 1 |   LIB$_FATERRLIB,    Fatal internal error
: 213      0470 1 |   LIB$_INVARG,       Invalid argument
: 214      0471 1 |   LIB$_NO_STREAM,    No stream active
: 215      0472 1 |   LIB$_INVSCRPOS,    Invalid screen coordinate position
: 216      0473 1 |   LIB$_SCRBUFOVF,    Screen buffer overflow
: 217      0474 1 |   LIB$_WRONUMARG;    Wrong number of arguments
: 218      0475 1 |
: 219      0476 1 | EXTERNAL
: 220      0477 1 |   SCR$AL_DEVDEPND2 : BLOCK [, BYTE];
: 221      0478 1 |
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

LIB\$DOWN_SCROLL - Down Scroll, Move Cursor up 0

J 15

16-Sep-1984 02:28:18

14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742

[VMSLIB.SRC]SCREEN.B32;1

Page 6

(3)

```

223 0479 1 $SBTTL 'LIB$DOWN_SCROLL - Down Scroll, Move Cursor up One Line'
224 0480 1 GLOBAL ROUTINE LIB$DOWN_SCROLL =
225 0481 1
226 0482 1 ++
227 0483 1 FUNCTIONAL DESCRIPTION:
228 0484 1
229 0485 1 This routine causes the cursor to be moved up 1 line to the
230 0486 1 same column of the previous line. If the cursor was on the
231 0487 1 top line to begin with, it stays where it was, but all the
232 0488 1 information on the screen appears to move down one line. The
233 0489 1 information that was on the bottom line is lost and a blank line
234 0490 1 appears at the top.
235 0491 1
236 0492 1 If a scrolling region is active ( on a VT100 or in mapping)
237 0493 1 then the logic above applies to the top and bottom lines of
238 0494 1 the scrolling region rather than the top and bottom line of
239 0495 1 the screen. If an UP_SCROLL is performed on the bottom line of
240 0496 1 the screen, or a DOWN_SCROLL is performed on the top line of the
241 0497 1 screen, while a scrolling region is active then no screen
242 0498 1 movement takes place unless the top or bottom line of the screen
243 0499 1 corresponds to the top or bottom line of the scrolling region.
244 0500 1
245 0501 1 CALLING SEQUENCE:
246 0502 1
247 0503 1 ret_status.wlc.v = LIB$DOWN_SCROLL ()
248 0504 1
249 0505 1 FORMAL PARAMETERS:
250 0506 1
251 0507 1 NONE
252 0508 1
253 0509 1 IMPLICIT INPUTS:
254 0510 1
255 0511 1 NONE
256 0512 1
257 0513 1 IMPLICIT OUTPUTS:
258 0514 1
259 0515 1 NONE
260 0516 1
261 0517 1 COMPLETION STATUS:
262 0518 1
263 0519 1 $$$_NORMAL Normal successful completion
264 0520 1
265 0521 1 SIDE EFFECTS:
266 0522 1
267 0523 1 NONE
268 0524 1 --
269 0525 1
270 0526 1 +
271 0527 1 Equate to SCR$ entry point.
272 0528 1 -
273 0529 1
274 0530 1 GLOBAL BIND ROUTINE LIB$DOWN_SCROLL = SCR$DOWN_SCROLL ;
275 0531 1 !<BLF/PAGE>
```



```
277 0532 1 %SBTTL 'LIB$ERASE_LINE - Erase Line from Screen'
278 0533 1 GLOBAL ROUTINE LIB$ERASE_LINE (
279 0534 1     LINE_NO : REF VECTOR [,WORD],
280 0535 1     COL_NO  : REF VECTOR [,WORD]
281 0536 1 ) =
282 0537 1 ++
283 0538 1 FUNCTIONAL DESCRIPTION:
284 0539 1
285 0540 1     This routine causes the screen to be erased from the specified
286 0541 1     position to the end of the line.  If the position is not
287 0542 1     specified, the current screen position is assumed.
288 0543 1
289 0544 1 CALLING SEQUENCE:
290 0545 1
291 0546 1     ret_status.wlc.v = LIB$ERASE_LINE ([LINE_NO.rw.r, COL_NO.rw.r])
292 0547 1
293 0548 1 FORMAL PARAMETERS:
294 0549 1
295 0550 1     LINE_NO.rw.r    Optional.  Address of line number where erasing
296 0551 1                   starts.  If this argument is omitted, COL_NO is
297 0552 1                   ignored as well, and current position will be
298 0553 1                   used.
299 0554 1
300 0555 1     COL_NO.rw.r     Optional.  Address of column number where
301 0556 1                   erasing starts.  If this argument is omitted,
302 0557 1                   LINE_NO is ignored as well, and current position
303 0558 1                   will be used.
304 0559 1
305 0560 1 IMPLICIT INPUTS:
306 0561 1
307 0562 1     NONE
308 0563 1
309 0564 1 IMPLICIT OUTPUTS:
310 0565 1
311 0566 1     NONE
312 0567 1
313 0568 1 COMPLETION STATUS:
314 0569 1
315 0570 1     $$$_NORMAL      Normal successful completion
316 0571 1
317 0572 1 SIDE EFFECTS:
318 0573 1
319 0574 1     NONE
320 0575 1 --
321 0576 1
322 0577 2 BEGIN
323 0578 2
324 0579 2 BUILTIN
325 0580 2     ACTUALCOUNT,
326 0581 2     ACTUALPARAMETER,
327 0582 2     AP,
328 0583 2     CALLG;
329 0584 2
330 0585 2 !+
331 0586 2 !- If no arguments, just call SCR$ERASE_LINE with original call list.
332 0587 2 !-
333 0588 2 IF ACTUALCOUNT() EQL 0
```

```

334 0589 2 THEN
335 0590 2 RETURN ( CALLG ( .AP, SCR$ERASE_LINE ) ) ;
336 0591 2
337 0592 2
338 0593 2 + Just 1 argument is a no-no. If LINE_NO is supplied, COL_NO must be
339 0594 2 supplied as well. Complain.
340 0595 2 -
341 0596 2 IF ACTUALCOUNT() EQL 1
342 0597 2 THEN
343 0598 2 IF ACTUALPARAMETER (1) EQL 0
344 0599 2 THEN RETURN ( SCR$ERASE_LINE ( ) ) ; 1 null arg - treat as no args
345 0600 2
346 0601 2 ELSE RETURN ( LIB$WRONUMARG ) ; 1 nonzero arg - error
347 0602 2
348 0603 2
349 0604 2 +
350 0605 2 Check supplied arguments. If they are not specified (arglist entry
351 0606 2 =0) then pass on a zero. If they are specified, check for a value
352 0607 2 of zero and complain. Call SCR$ERASE_LINE with appropriate
353 0608 2 combination of arguments promoted to by-value.
354 0609 2 -
355 0610 2 IF ACTUALCOUNT() EQL 2
356 0611 2 THEN
357 0612 2 RETURN ( SCR$ERASE_LINE (
358 0613 2 IF ACTUALPARAMETER(1) NEQ 0
359 0614 2 THEN
360 0615 2 IF .LINE_NO[0] EQL 0
361 0616 2 THEN
362 0617 2 RETURN (LIB$INVSCRPOS)
363 0618 2 ELSE
364 0619 2 .LINE_NO[0]
365 0620 2 ELSE 0,
366 0621 2
367 0622 2 IF ACTUALPARAMETER(2) NEQ 0
368 0623 2 THEN
369 0624 2 IF .COL_NO[0] EQL 0
370 0625 2 THEN
371 0626 2 RETURN (LIB$INVSCRPOS)
372 0627 2 ELSE .COL_NO[0]
373 0628 2 ELSE 0 ) )
374 0629 2
375 0630 2 ELSE
376 0631 2 RETURN ( LIB$WRONUMARG ) ;
377 0632 1 END; ! End of routine LIB$ERASE_LINE
```

```
.TITLE LIB$SCREEN LIB$SCREEN - Screen Management
.IDENT \V04-000\
```

```
.PSECT _LIB$DATA,NOEXE, PIC.2
```

```
00000000 00000 SCR$ _FLINKHEAD::
               .LONG 0
00000000 00004 SCR$ _CUROUTPUT::
               .LONG 0
00000000 00008 SCR$ _WORKMASK:
               .LONG 0
```

				.EXTRN	LIB\$ANALYZE_SDESC R2	
				.EXTRN	LIB\$ASSIGN, LIB\$CALL IMAGE	
				.EXTRN	LIB\$FREE EF, LIB\$FREE VM	
				.EXTRN	LIB\$GET EF, LIB\$GET INPUT	
				.EXTRN	LIB\$GET VM, LIB\$LP LINES	
				.EXTRN	LIB\$PUT OUTPUT, LIB\$FREE1 DD	
				.EXTRN	LIB\$SIG TO RET, SCR\$SET OUTPUT	
				.EXTRN	SCR\$STOP OUTPUT	
				.EXTRN	STR\$CONCAT, STR\$DUPL CHAR	
				.EXTRN	LIB\$ FATERRLIB, LIB\$ INVARG	
				.EXTRN	LIB\$ NO STRACT, LIB\$ INVSCRPOS	
				.EXTRN	LIB\$ SCRBUFOVF, LIB\$ WRONUMARG	
				.EXTRN	SCR\$AL_DEVDEPND2	
				.PSECT	_LIB\$CODE, NOWRT, SHR, PIC, 2	
				.ENTRY	LIB\$ERASE_LINE, Save R2	: 0533
				MOVAB	SCR\$ERASE_LINE, R2	
				TSTB	(AP)	: 0588
				BNEQ	1\$	
				CALLG	(AP), SCR\$ERASE_LINE	: 0590
				RET		
				CMPB	(AP), #1	: 0596
				BNEQ	2\$	
				TSTL	4(AP)	: 0598
				BNEQ	9\$	
				CALLS	#0, SCR\$ERASE_LINE	: 0600
				RET		: 0602
				CMPB	(AP), #2	: 0610
				BNEQ	9\$	
				TSTL	8(AP)	: 0622
				BEQL	3\$	
				TSTW	@COL_NO	: 0624
				BEQL	5\$	
				MOVZWL	@COL_NO, R0	: 0627
				PUSHL	R0	: 0624
				BRB	4\$	
				CLRL	-(SP)	: 0622
				TSTL	4(AP)	: 0613
				BEQL	7\$	
				TSTW	@LINE_NO	: 0615
				BNEQ	6\$	
				MOVL	#LIB\$ INVSCRPOS, R0	: 0617
				RET		
				MOVZWL	@LINE_NO, R0	: 0619
				PUSHL	R0	: 0615
				BRB	8\$	
				CLRL	-(SP)	: 0613
				CALLS	#2, SCR\$ERASE_LINE	: 0631
				RET		
				MOVL	#LIB\$ WRONUMARG, R0	: 0632
				RET		

52	0000V	CF	9E	00002	
		6C	95	00007	
		04	12	00009	
62		6C	FA	0000B	
			04	0000E	
01		6C	91	0000F	1\$:
		09	2	00012	
	04	AC	D5	00014	
		3D	12	00017	
62		00	FB	00019	
			04	0001C	
02		6C	91	0001D	2\$:
		34	12	00020	
	08	AC	D5	00022	
		0D	13	00025	
	08	BC	B5	00027	
		14	13	0002A	
50	08	BC	3C	0002C	
		50	DD	00030	
		02	11	00032	
		7E	D4	00034	3\$:
	04	AC	D5	00036	4\$:
		15	13	00039	
	04	BC	B5	0003B	
		08	12	0003E	
50	00000000G	8F	D0	00040	5\$:
			04	00047	
50	04	BC	3C	00048	6\$:
		50	DD	0004C	
		02	11	0004E	
		7E	D4	00050	7\$:
62		02	FB	00052	8\$:
			04	00055	
50	00000000G	8F	D0	00056	9\$:
			04	0005D	

; Routine Size: 94 bytes, Routine Base: _LIB\$CODE + 0000

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
LIB\$ERASE_LINE - Erase Line from Screen

: 378

0633 1 !<BLF/PAGE>

N 15
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 10
(4)

```
380 0634 1 %SBTTL 'LIB$ERASE_PAGE - Erase Page from Screen'
381 0635 1 GLOBAL ROUTINE LIB$ERASE_PAGE (
382 0636 1 LINE_NO : REF VECTOR [,WORD],
383 0637 1 COL_NO : REF VECTOR [,WORD]
384 0638 1 ) =
385 0639 1 ++
386 0640 1 FUNCTIONAL DESCRIPTION:
387 0641 1
388 0642 1 This routine causes the screen to be erased from the specified
389 0643 1 position to the end of the screen. If the position is not
390 0644 1 specified, the current screen position is assumed.
391 0645 1
392 0646 1 CALLING SEQUENCE:
393 0647 1
394 0648 1 ret_status.wlc.v = LIB$ERASE_PAGE ([LINE_NO.rw.r, COL_NO.rw.r])
395 0649 1
396 0650 1 FORMAL PARAMETERS:
397 0651 1
398 0652 1 LINE_NO.rw.r Optional. Address of line number at which to
399 0653 1 start erasing. If omitted, COL_NO will be
400 0654 1 ignored as well and current cursor position
401 0655 1 will be used.
402 0656 1
403 0657 1 COL_NO.rw.r Optional. Address of Column number at which
404 0658 1 to start erasing. If omitted, LINE_NO will be
405 0659 1 ignored as well and current cursor position
406 0660 1 will be used.
407 0661 1
408 0662 1 IMPLICIT INPUTS:
409 0663 1
410 0664 1 NONE
411 0665 1
412 0666 1 IMPLICIT OUTPUTS:
413 0667 1
414 0668 1 NONE
415 0669 1
416 0670 1 COMPLETION STATUS:
417 0671 1
418 0672 1 $$$_NORMAL Normal successful completion
419 0673 1
420 0674 1 SIDE EFFECTS:
421 0675 1
422 0676 1 NONE
423 0677 1 --
424 0678 1
425 0679 2 BEGIN
426 0680 2
427 0681 2 BUILTIN
428 0682 2 ACTUALCOUNT,
429 0683 2 ACTUALPARAMETER,
430 0684 2 AP,
431 0685 2 CALLG;
432 0686 2
433 0687 2 +
434 0688 2 If no arguments, just call SCR$ERASE_PAGE with original call list.
435 0689 2 -
436 0690 2 IF ACTUALCOUNT() EQL 0
```

```

: 437 0691 2 THEN
: 438 0692 2 RETURN ( CALLG ( .AP, SCR$ERASE_PAGE ) ) ;
: 439 0693 2
: 440 0694 2 !+
: 441 0695 2 Just 1 argument is a no-no. If LINE_NO is supplied, COL_NO must be
: 442 0696 2 supplied as well. Complain.
: 443 0697 2 !-
: 444 0698 2 IF ACTUALCOUNT() EQL 1
: 445 0699 2 THEN
: 446 0700 2 IF ACTUALPARAMETER (1) EQL 0
: 447 0701 2 THEN ! 1 null arg - treat as no args
: 448 0702 3 RETURN ( SCR$ERASE_PAGE ( ) )
: 449 0703 2 ELSE ! 1 nonzero arg - error
: 450 0704 2 RETURN ( LIB$_WRONUMARG ) ;
: 451 0705 2
: 452 0706 2 !+
: 453 0707 2 Check supplied arguments. If they are not specified (arglist entry
: 454 0708 2 =0) then pass on a zero. If they are specified, check for a value
: 455 0709 2 of zero and complain. Call SCR$ERASE_PAGE with appropriate
: 456 0710 2 combination of arguments promoted to by-value.
: 457 0711 2 !-
: 458 0712 2 IF ACTUALCOUNT() EQL 2
: 459 0713 2 THEN
: 460 0714 3 RETURN ( SCR$ERASE_PAGE (
: 461 0715 3 IF ACTUALPARAMETER(1) NEQ 0
: 462 0716 3 THEN
: 463 0717 3 IF .LINE_NO[0] EQL 0
: 464 0718 3 THEN
: 465 0719 4 RETURN (LIB$_INVSCRPOS)
: 466 0720 3 ELSE
: 467 0721 3 .LINE_NO[0]
: 468 0722 3 ELSE 0,
: 469 0723 3
: 470 0724 3 IF ACTUALPARAMETER(2) NEQ 0
: 471 0725 3 THEN
: 472 0726 3 IF .COL_NO[0] EQL 0
: 473 0727 3 THEN
: 474 0728 4 RETURN (LIB$_INVSCRPOS)
: 475 0729 3 ELSE .COL_NO[0]
: 476 0730 3 ELSE 0 ) )
: 477 0731 3
: 478 0732 2 ELSE
: 479 0733 2 RETURN ( LIB$_WRONUMARG ) ;
: 480 0734 1 END; ! End of routine LIB$ERASE_PAGE
```

		0004 00000	.ENTRY	LIB\$ERASE_PAGE, Save R2	: 0635
52	0000V	CF 9E 00002	MOVAB	SCR\$ERASE_PAGE, R2	: 0690
		6C 95 00007	TSTB	(AP)	: 0692
		04 12 00009	BNEQ	1\$	: 0698
62		6C FA 0000B	CALLG	(AP), SCR\$ERASE_PAGE	
		04 0000E	RET		
01		6C 91 0000F 1\$:	CMPB	(AP), #1	
		09 12 00012	BNEQ	2\$	

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
LIB\$ERASE_PAGE - Erase Page from Screen

D 16
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 13
(5)

	04	AC	D5	00014	TSTL	4(AP)	: 0700
		3D	12	00017	BNEQ	9\$	: :
62		00	FB	00019	CALLS	#0, SCR\$ERASE_PAGE	: 0702
			04	0001C	RET		: 0704
02		6C	91	0001D	2\$: CMPB	(AP), #2	: 0712
		34	12	00020	BNEQ	9\$	: :
	08	AC	D5	00022	TSTL	8(AP)	: 0724
		0D	13	00025	BEQL	3\$	: :
	08	BC	B5	00027	TSTW	@COL_NO	: 0726
		14	13	0002A	BEQL	5\$	: :
50	08	BC	3C	0002C	MOVZWL	@COL_NO, R0	: 0729
		50	DD	00030	PUSHL	R0	: 0726
		02	11	00032	BRB	4\$	: :
		7E	D4	00034	3\$: CLRL	-(SP)	: 0724
	04	AC	D5	00036	4\$: TSTL	4(AP)	: 0715
		15	13	00039	BEQL	7\$	: :
	04	BC	B5	0003B	TSTW	@LINE_NO	: 0717
		08	12	0003E	BNEQ	6\$	: :
50	00000000G	8F	D0	00040	5\$: MOVL	#LIB\$_INVSCRPOS, R0	: 0719
			04	00047	RET		: :
50	04	BC	3C	00048	6\$: MOVZWL	@LINE_NO, R0	: 0721
		50	DD	0004C	PUSHL	R0	: 0717
		02	11	0004E	BRB	8\$	: :
		7E	D4	00050	7\$: CLRL	-(SP)	: 0715
62		02	FB	00052	8\$: CALLS	#2, SCR\$ERASE_PAGE	: :
			04	00055	RET		: 0733
50	00000000G	8F	D0	00056	9\$: MOVL	#LIB\$_WRONUMARG, R0	: :
			04	0005D	RET		: 0734

; Routine Size: 94 bytes, Routine Base: _LIB\$CODE + 005E

; 481 0735 1 !<BLF/PAGE>

```

483 0736 1 %SBTTL 'LIB$PUT_BUFFER - Put Current Buffer to Screen or to Prev. Buffer'
484 0737 1 GLOBAL ROUTINE [LIB$PUT_BUFFER] (
485 0738 1     OLD_BUFFER : REF VECTOR [,LONG]
486 0739 1 ) =
487 0740 1
488 0741 1 ++
489 0742 1 FUNCTIONAL DESCRIPTION:
490 0743 1     This routine is the converse of LIB$SET_BUFFER. It puts the
491 0744 1     contents of the current buffer to another buffer or the screen
492 0745 1     if no buffer is given.
493 0746 1
494 0747 1 CALLING SEQUENCE:
495 0748 1
496 0749 1     ret_status.wlc.v = LIB$PUT_BUFFER ([OLD_BUFFER.rl.r])
497 0750 1
498 0751 1 FORMAL PARAMETERS:
499 0752 1
500 0753 1     OLD_BUFFER.rl.r     Optional. Address of the previous
501 0754 1                      buffer to put the current buffer into.
502 0755 1                      If not specified or zero, put current
503 0756 1                      buffer to the screen.
504 0757 1
505 0758 1 IMPLICIT INPUTS:
506 0759 1
507 0760 1     NONE
508 0761 1
509 0762 1 IMPLICIT OUTPUTS:
510 0763 1
511 0764 1     NONE
512 0765 1
513 0766 1 COMPLETION STATUS:
514 0767 1
515 0768 1     SS$_NORMAL         Normal successful completion
516 0769 1
517 0770 1 SIDE EFFECTS:
518 0771 1
519 0772 1     The current buffer is output and reinitialized to empty. If
520 0773 1     OLD_BUFFER is nonzero, it is established to become the new
521 0774 1     current buffer.
522 0775 1 --
523 0776 1
524 0777 2 BEGIN
525 0778 2
526 0779 2 BUILTIN
527 0780 2     ACTUALCOUNT,
528 0781 2     ACTUALPARAMETER;
529 0782 2
530 0783 2 ++
531 0784 2 If no arguments supplied to us, just call SCR$PUT_BUFFER with no
532 0785 2 arguments.
533 0786 2 --
534 0787 2 IF ACTUALCOUNT() EQL 0
535 0788 2 THEN
536 0789 2     RETURN ( SCR$PUT_BUFFER() );
537 0790 2
538 0791 2 ++
539 0792 2 If parameter supplied, and it isn't zero, pass it on to SCR$PUT_BUFFER
```


LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

LIB\$PUT_BUFFER - Put Current Buffer to Screen o

F 16

16-Sep-1984 02:28:18

VAX-11 Bliss-32 V4.0-742

[VMSLIB.SRC]SCREEN.B32;1

Page 15

(6)

```
: 540      0793 2 ! with the by-value address. If parameter is zero, call SCR$PUT_BUFFER
: 541      0794 2 ! with no arguments.
: 542      0795 2 !-
: 543      0796 2     IF ACTUALCOUNT() EQL 1
: 544      0797 2     THEN
: 545      0798 2         IF ACTUALPARAMETER(1) NEQ 0
: 546      0799 2         THEN
: 547      0800 3             RETURN ( SCR$PUT_BUFFER ( OLD_BUFFER[0] ) )
: 548      0801 2         ELSE
: 549      0802 3             RETURN ( SCR$PUT_BUFFER ( ) )
: 550      0803 2     ELSE
: 551      0804 2         RETURN ( LIB$_WRONUMARG ) ;
: 552      0805 2
: 553      0806 1     END;                                ! End of routine LIB$PUT_BUFFER
```

			0000	00000	.ENTRY	LIB\$PUT_BUFFER, Save nothing	: 0737
			6C	95 00C02	TSTB	(AP)	: 0787
			13	13 00004	BEQL	1\$	
	01		6C	91 00006	CMPB	(AP), #1	: 0796
			14	12 00009	BNEQ	2\$	
		04	AC	D5 0300B	TSTL	4(AP)	: 0798
			09	13 0000E	BEQL	1\$	
		04	AC	DD 00010	PUSHL	OLD_BUFFER	: 0800
0000V	CF		01	FB 00013	CALLS	#1, SCR\$PUT_BUFFER	
				04 00018	RET		
0000V	CF		00	FB 00019 1\$:	CALLS	#0, SCR\$PUT_BUFFER	: 0802
				04 0001E	RET		: 0804
	50 00000000G	8F	D0	0001F 2\$:	MOVL	#LIB\$_WRONUMARG, R0	
			04	00026	RET		: 0806

; Routine Size: 39 bytes, Routine Base: _LIB\$CODE + 00BC

: 554 0807 1 !<BLF/PAGE>

```

556 0808 1 %SBTTL 'LIB$PUT_LINE - Put Text to Screen in Line Mode'
557 0809 1 GLOBAL ROUTINE [LIB$PUT_LINE (
558 0810 1 TEXT : REF BLOCK [,BYTE],
559 0811 1 LINE_ADV,
560 0812 1 FLAGS
561 0813 1 ) =
562 0814 1
563 0815 1 ++
564 0816 1 FUNCTIONAL DESCRIPTION:
565 0817 1
566 0818 1 This routine is used to write messages to the terminal
567 0819 1 followed by cursor movement sequences.
568 0820 1
569 0821 1 CALLING SEQUENCE:
570 0822 1
571 0823 1 ret_status.wlc.v = LIB$PUT_LINE (TEXT.rt.dx
572 0824 1 [,LINE_ADV.rl.r]
573 0825 1 [,FLAGS.rl.r])
574 0826 1
575 0827 1 FORMAL PARAMETERS:
576 0828 1
577 0829 1 TEXT.rt.dx Address of descriptor of output string.
578 0830 1
579 0831 1 LINE_ADV.rl.r Optional. Address of signed number of lines
580 0832 1 to advance after output.
581 0833 1
582 0834 1 FLAGS.rl.r Optional. Rendition codes.
583 0835 1 If omitted, SCR$M_NORMAL will be used.
584 0836 1 Values:
585 0837 1 SCR$M_BLINK display characters blinking.
586 0838 1 SCR$M_BOLD display characters in
587 0839 1 higher-than-normal intensity.
588 0840 1 SCR$M_NORMAL display characters using
589 0841 1 rendition associated with
590 0842 1 window.
591 0843 1 SCR$M_REVERSE display characters in reverse
592 0844 1 video -- i.e., using opposite
593 0845 1 rendition from window default.
594 0846 1 SCR$M_UNDERLINE display characters underlined.
595 0847 1
596 0848 1 IMPLICIT INPUTS:
597 0849 1
598 0850 1 NONE
599 0851 1
600 0852 1 IMPLICIT OUTPUTS:
601 0853 1
602 0854 1 NONE
603 0855 1
604 0856 1 COMPLETION STATUS:
605 0857 1
606 0858 1 SS$ _NORMAL Normal successful completion
607 0859 1
608 0860 1 SIDE EFFECTS:
609 0861 1
610 0862 1 NONE
611 0863 1 --
612 0864 1
```

```

: 613      0865 2   BEGIN
: 614      0866 2
: 615      0867 2   BUILTIN
: 616      0868 2       ACTUALCOUNT,
: 617      0869 2       ACTUALPARAMETER ;
: 618      0870 2
: 619      0871 2   LITERAL
: 620      0872 2       K_ADV_ARG = 2;
: 621      0873 2       K_FLAG_ARG = 3;
: 622      0874 2
: 623      0875 2   LOCAL
: 624      0876 2       STATUS;
: 625      0877 2
: 626      0878 2   IF ACTUALCOUNT () NEQ 0
: 627      0879 2   THEN
: 628      0880 3       BEGIN
: 629      0881 4       RETURN (STATUS = SCR$PUT_LINE (.TEXT,
: 630      0882 5           (IF ACTUALCOUNT () GEQ K_ADV_ARG AND
: 631      0883 5           ACTUALPARAMETER (K_ADV_ARG) NEQ 0
: 632      0884 5           THEN .LINE_ADV
: 633      0885 4           ELSE 1),
: 634      0886 5           (IF ACTUALCOUNT () GEQ K_FLAG_ARG AND
: 635      0887 5           ACTUALPARAMETER (K_FLAG_ARG) NEQ 0
: 636      0888 5           THEN .FLAGS
: 637      0889 5           ELSE SCR$M_NORMAL)
: 638      0890 3       ));
: 639      0891 3       END
: 640      0892 2   ELSE
: 641      0893 2       RETURN (LIB$_WRONUMARG);      ! error if no arguments
: 642      0894 2
: 643      0895 2
: 644      0896 1   END;                          ! End of routine LIB$PUT_LINE
```

		0000	00000	.ENTRY	LIB\$PUT_LINE, Save nothing	: 0809	
		6C	95 00002	TSTB	(AP)	: 0878	
		2B	13 00004	BEQL	5\$		
03		6C	91 00006	CMPB	(AP), #3	: 0886	
		0A	1F 00009	BLSSU	1\$		
	0C	AC	D5 0000B	TSTL	12(AP)	: 0887	
		05	13 0000E	BEQL	1\$		
	0C	BC	DD 00010	PUSHL	@FLAGS	: 0888	
		02	11 00013	BRB	2\$		
		7E	D4 00015	CLRL	-(SP)	: 0886	
02		6C	91 00017	CMPB	(AP), #2	: 0882	
		0A	1F 0001A	BLSSU	3\$		
	08	AC	D5 0001C	TSTL	8(AP)	: 0883	
		05	13 0001F	BEQL	3\$		
	08	BC	DD 00021	PUSHL	@LINE_ADV	: 0884	
		02	11 00024	BRB	4\$		
		01	DD 00026	PUSHL	#1	: 0882	
		04	AC	DD 00028	PUSHL	TEXT	: 0881
0000V	CF	03	FB 0002B	CALLS	#3, SCR\$PUT_LINE		
		04	00030	RET		: 0893	

LIB\$SCREEN	LIB\$SCREEN - Screen Management	I 16		
V04-000	LIB\$PUT_LINE - Put Text to Screen in Line Mode	16-Sep-1984 02:28:18	VAX-11 Bliss-32 V4.0-742	Page 18
		14-Sep-1984 13:34:42	[VMSLIB.SRC]SCREEN.B32;1	(7)
	50 00000000G 8F D0 00031 5\$:	MOVL	#LIB\$_WRONUMARG, R0	
	04 00038	RET		: 0896
; Routine Size: 57 bytes, Routine Base: _LIB\$CODE + 00E3				
; 645 0897 1 !<BLF/PAGE>				

```

647 0898 1 %SBTTL 'LIB$PUT_SCREEN - Put Text to Screen'
648 0899 1 GLOBAL ROUTINE LIB$PUT_SCREEN (
649 0900 1     TEXT      : REF BLOCK [ BYTE],
650 0901 1     LINE_NO : REF VECTOR [ WORD],
651 0902 1     COL_NO  : REF VECTOR [ WORD],
652 0903 1     FLAGS   : REF VECTOR [ LONG]
653 0904 1 ) =
654 0905 1
655 0906 1 ++
656 0907 1 FUNCTIONAL DESCRIPTION:
657 0908 1
658 0909 1     This routine is used to write messages to the terminal.
659 0910 1
660 0911 1 CALLING SEQUENCE:
661 0912 1
662 0913 1     ret_status.wlc.v = LIB$PUT_SCREEN (TEXT.rt.dx
663 0914 1                               [,LINE_NO.rl.r, COL_NO.rl.r]
664 0915 1                               [,FLAGS.rl.r])
665 0916 1
666 0917 1 FORMAL PARAMETERS:
667 0918 1
668 0919 1     TEXT.rt.dx      Address of descriptor of output string.
669 0920 1
670 0921 1     LINE_NO.rl.r    Optional. Address of line number at which to
671 0922 1                    start output. If omitted (=0), the current
672 0923 1                    line number is used.
673 0924 1
674 0925 1     COL_NO.rl.r     Optional. Address of column number at which
675 0926 1                    to start output. If omitted (=0), the current
676 0927 1                    line number is used.
677 0928 1
678 0929 1     FLAGS.rl.r      Optional. Rendition codes.
679 0930 1                    If omitted, SCR$M_NORMAL will be used.
680 0931 1                    Values:
681 0932 1                    SCR$M_BLINK      display characters blinking.
682 0933 1                    SCR$M_BOLD       display characters in
683 0934 1                    higher-than-normal intensity.
684 0935 1                    SCR$M_NORMAL     display characters using
685 0936 1                    rendition associated with
686 0937 1                    window.
687 0938 1                    SCR$M_REVERSE    display characters in reverse
688 0939 1                    video -- i.e., using opposite
689 0940 1                    rendition from window default.
690 0941 1                    SCR$M_UNDERLINE  display characters underlined.
691 0942 1
692 0943 1 IMPLICIT INPUTS:
693 0944 1
694 0945 1     NONE
695 0946 1
696 0947 1 IMPLICIT OUTPUTS:
697 0948 1
698 0949 1     NONE
699 0950 1
700 0951 1 COMPLETION STATUS:
701 0952 1
702 0953 1     $$$ NORMAL      Normal successful completion
703 0954 1     LIB$_INVARG
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
LIB\$PUT_SCREEN - Put Text to Screen

K 16
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 20
(8)

```

: 704      0955 1 |      LIB$_INVSCRPOS
: 705      0956 1 |
: 706      0957 1 |      SIDE EFFECTS:
: 707      0958 1 |
: 708      0959 1 |      NONE
: 709      0960 1 |      --
: 710      0961 1 |
: 711      0962 2 |      BEGIN
: 712      0963 2 |
: 713      0964 2 |      BUILTIN
: 714      0965 2 |          ACTUALCOUNT,
: 715      0966 2 |          ACTUALPARAMETER,
: 716      0967 2 |          AP,
: 717      0968 2 |          CALLG ;
: 718      0969 2 |
: 719      0970 2 |      CASE ACTUALCOUNT() FROM 1 TO 4 OF
: 720      0971 2 |      SET
: 721      0972 2 |          +
: 722      0973 2 |          Only 1 argument (TEXT), call SCR$PUT_SCREEN with existing
: 723      0974 2 |          call list.
: 724      0975 2 |          -
: 725      0976 2 |          [1]:
: 726      0977 2 |              RETURN ( CALLG ( .AP, SCR$PUT_SCREEN ) ) ;
: 727      0978 2 |
: 728      0979 2 |          +
: 729      0980 2 |          Two arguments. This is a no-no. If LINE_NO is specified,
: 730      0981 2 |          COL_NO must be specified as well.
: 731      0982 2 |          -
: 732      0983 2 |          [2]:
: 733      0984 2 |              RETURN (LIB$_WRONUMARG) ;
: 734      0985 2 |
: 735      0986 2 |          +
: 736      0987 2 |          Three arguments provided. Must determine which (if any)
: 737      0988 2 |          were not specified (call list entry is zero). For specified
: 738      0989 2 |          LINE_NO and COL_NO, check for values of zero. Call
: 739      0990 2 |          SCR$PUT_SCREEN with appropriate combination of arguments
: 740      0991 2 |          promoted to by-value.
: 741      0992 2 |          -
: 742      0993 2 |          [3]:
: 743      0994 2 |              RETURN
: 744      0995 2 |                  ( SCR$PUT_SCREEN ( .TEXT,
: 745      0996 2 |                      IF ACTUALPARAMETER(2) NEQ 0
: 746      0997 2 |                      THEN
: 747      0998 2 |                          IF .LINE_NO[0] EQL 0
: 748      0999 2 |                          THEN
: 749      1000 2 |                              RETURN (LIB$_INVSCRPOS)
: 750      1001 2 |                              ELSE
: 751      1002 2 |                                  .LINE_NO[0]
: 752      1003 2 |                              ELSE 0,
: 753      1004 2 |                      IF ACTUALPARAMETER(3) NEQ 0
: 754      1005 2 |                      THEN
: 755      1006 2 |                          IF .COL_NO[0] EQL 0
: 756      1007 2 |                          THEN
: 757      1008 2 |                              RETURN (LIB$_INVSCRPOS)
: 758      1009 2 |                              ELSE .COL_NO[0]
: 759      1010 2 |                      ELSE 0 ) );
: 760      1011 2 |

```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
LIB\$PUT_SCREEN - Put Text to Screen

L 16
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 21
(8)

```

: 761      1012  2
: 762      1013  2
: 763      1014  2
: 764      1015  2
: 765      1016  2
: 766      1017  2
: 767      1018  2
: 768      1019  2
: 769      1020  2
: 770      1021  2
: 771      1022  2
: 772      1023  2
: 773      1024  2
: 774      1025  2
: 775      1026  2
: 776      1027  2
: 777      1028  2
: 778      1029  2
: 779      1030  2
: 780      1031  2
: 781      1032  2
: 782      1033  2
: 783      1034  2
: 784      1035  2
: 785      1036  2
: 786      1037  2
: 787      1038  2
: 788      1039  2
: 789      1040  2
: 790      1041  2
: 791      1042  2
: 792      1043  2
: 793      1044  2
: 794      1045  2
: 795      1046  2
: 796      1047  2
: 797      1048  2
: 798      1049  2
: 799      1050  2
: 800      1051  2
: 801      1052  2
: 802      1053  1

+
Four arguments provided. Must determine which (if any)
were not specified (call list entry is zero). For specified
LINE_NO and COL_NO, check for values of zero. Call
SCR$PUT_SCREEN with appropriate combination of arguments
promoted to by-value.
[4]:
    RETURN
      ( SCR$PUT_SCREEN ( .TEXT,
        IF ACTUALPARAMETER(2) NEQ 0
        THEN
          IF .LINE_NO[0] EQL 0
          THEN
            RETURN (LIB$_INVSCRPOS)
          ELSE
            .LINE_NO[0]
        ELSE 0,
        IF ACTUALPARAMETER(3) NEQ 0
        THEN
          IF .COL_NO[0] EQL 0
          THEN
            RETURN (LIB$_INVSCRPOS)
          ELSE .COL_NO[0]
        ELSE 0,
        IF ACTUALPARAMETER(4) NEQ 0
        THEN
          .FLAGS[0]
        ELSE
          0 )) ;

+
Too many ( or no) arguments provided. Complain.
[OUTRANGE]:
    RETURN ( LIB$_WRONUMARG ) ;

TES ;
END;                                     ! End of routine LIB$PUT_SCREEN
```

0045	03	52	0000V	CF	9E	00002	.ENTRY	LIB\$PUT_SCREEN, Save R2	: 0899
	C016	01		6C	8F	00007	MOVAB	SCR\$PUT_SCREEN, R2	: 1021
		000E	000A			0000B	CASEB	(AP), #T, #3	: 1021
							.WORD	2\$-1\$,-	: 1021
								3\$-1\$,-	: 1021
								4\$-1\$,-	: 1021
								9\$-1\$,-	: 1021
								3\$	: 1050
		62	04	11	00013		BRB		: 0977
			6C	FA	00015	2\$:	CALLG	(AP), SCR\$PUT_SCREEN	: 1021
				04	00018		RET		: 1021

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
LIB\$PUT_SCREEN - Put Text to Screen

M 16

16-Sep-1984 02:28:18

14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742

[VMSLIB.SRC]SCREEN.B32;1

Page 22
(8)

50	00000000G	8F	D0	00019	3\$:	MOVL	#LIB\$_WRCNUMARG, R0	:	
			04	00020		RET		:	
	0C	AC	D5	00021	4\$:	TSTL	12(AP)	:	1005
		0D	13	00024		BEQL	5\$	:	
	0C	BC	B5	00026		TSTW	@COL_NO	:	1007
		4F	13	00029		BEQL	14\$	:	
50	0C	BC	3C	0002B		MOVZWL	@COL_NO, R0	:	1010
		50	DD	0002F		PUSHL	R0	:	1007
		02	11	00031		BRB	6\$	:	
		7E	D4	00033	5\$:	CLRL	-(SP)	:	1005
	08	AC	D5	00035	6\$:	TSTL	8(AP)	:	0996
		0D	13	00038		BEQL	7\$	:	
	08	BC	B5	0003A		TSTW	@LINE_NO	:	0998
		3B	13	0003D		BEQL	14\$	:	
50	08	BC	3C	0003F		MOVZWL	@LINE_NO, R0	:	1002
		50	DD	00043		PUSHL	R0	:	0998
		02	11	00045		BRB	8\$	:	
		7F	D4	00047	7\$:	CLRL	-(SP)	:	0996
	04	AC	DD	00049	8\$:	PUSHL	TEXT	:	0995
62		03	FB	0004C		CALLS	#3, SCR\$PUT_SCREEN	:	
			04	0004F		RET		:	1021
	10	AC	D5	00050	9\$:	TSTL	16(AP)	:	1040
		05	13	00053		BEQL	10\$	:	
	10	BC	DD	00055		PUSHL	@FLAGS	:	1042
		02	11	00058		BRB	11\$	:	
		7E	D4	0005A	10\$:	CLRL	-(SP)	:	1040
	0C	AC	D5	0005C	11\$:	TSTL	12(AP)	:	1032
		0D	13	0005F		BEQL	12\$	:	
	0C	BC	B5	00061		TSTW	@COL_NO	:	1034
		14	13	00064		BEQL	14\$	:	
50	0C	BC	3C	00066		MOVZWL	@COL_NO, R0	:	1037
		50	DD	0006A		PUSHL	R0	:	1034
		02	11	0006C		BRB	13\$	:	
		7E	D4	0006E	12\$:	CLRL	-(SP)	:	1032
	08	AC	D5	00070	13\$:	TSTL	8(AP)	:	1023
		15	13	00073		BEQL	16\$	:	
	08	BC	B5	00075		TSTW	@LINE_NO	:	1025
		08	12	00078		BNEQ	15\$	:	
50	00000000G	8F	D0	0007A	14\$:	MOVL	#LIB\$_INVSCRPOS, R0	:	1027
			04	00081		RET		:	
50	08	BC	3C	00082	15\$:	MOVZWL	@LINE_NO, R0	:	1029
		50	DD	00086		PUSHL	R0	:	1025
		02	11	00088		BRB	17\$	:	
		7E	D4	0008A	16\$:	CLRL	-(SP)	:	1023
	04	AC	DD	0008C	17\$:	PUSHL	TEXT	:	1022
62		04	FB	0008F		CALLS	#4, SCR\$PUT_SCREEN	:	
			04	00092		RET		:	1053

; Routine Size: 147 bytes, Routine Base: _LIB\$CODE + 011C

; 803 1054 1 !<BLF/PAGE>


```

805 1055 1 *SBTTL 'LIB$SET_BUFFER - Set or Clear Screen Buffer Mode'
806 1056 1 GLOBAL ROUTINE LIB$SET_BUFFER (
807 1057 1     BUFFER      : REF BLOCK [,BYTE],
808 1058 1     OLD_BUFFER : REF VECTOR [,LONG]
809 1059 1 ) =
810 1060 1 ++
811 1061 1 FUNCTIONAL DESCRIPTION:
812 1062 1
813 1063 1     This routine is called to establish a buffer to be used to hold
814 1064 1     terminal output rather than immediately writing the output to
815 1065 1     SYSS$OUTPUT. It is also called to turn off the buffering mode.
816 1066 1     The buffer address is established such that all screen output
817 1067 1     is appended to the end of the buffer.
818 1068 1
819 1069 1 CALLING SEQUENCE:
820 1070 1
821 1071 1     ret_status.wlc.v = LIB$SET_BUFFER (BUFFER.mt.ds
822 1072 1                                     [,OLD_BUFFER.wl.r])
823 1073 1
824 1074 1 FORMAL PARAMETERS:
825 1075 1
826 1076 1     BUFFER.mt.ds    Address of descriptor of buffer. If 0,
827 1077 1                    buffering mode is turned off.
828 1078 1
829 1079 1     OLD_BUFFER.wl.r Optional. Address of the location containing
830 1080 1                    address of the current buffer descriptor.
831 1081 1
832 1082 1 IMPLICIT INPUTS:
833 1083 1
834 1084 1     NONE
835 1085 1
836 1086 1 IMPLICIT OUTPUTS:
837 1087 1
838 1088 1     NONE
839 1089 1
840 1090 1 COMPLETION STATUS:
841 1091 1
842 1092 1     SS$NORMAL      Normal successful completion
843 1093 1
844 1094 1 SIDE EFFECTS:
845 1095 1
846 1096 1     NONE
847 1097 1 --
848 1098 1
849 1099 1 +
850 1100 1 Equate to SCR$ entry point.
851 1101 1
852 1102 1 GLOBAL BIND ROUTINE LIB$SET_BUFFER = SCR$SET_BUFFER ;
853 1103 1 !<BLF/PAGE>
```

```
855 1104 1 %SBTTL 'LIB$SET_CURSOR - Set Cursor to Character Position on Screen'
856 1105 1 GLOBAL ROUTINE [LIB$SET_CURSOR] (
857 1106 1     LINE_NO : REF VECTOR [,WORD],
858 1107 1     COL_NO  : REF VECTOR [,WORD]
859 1108 1 ) =
860 1109 1 --
861 1110 1 ++
862 1111 1 FUNCTIONAL DESCRIPTION:
863 1112 1     This routine causes the cursor to be moved to the specified
864 1113 1     position.
865 1114 1
866 1115 1 CALLING SEQUENCE:
867 1116 1
868 1117 1     ret_status.wlc.v = LIB$SET_CURSOR (LINE_NO.rw.r,
869 1118 1     COL_NO.rw.r)
870 1119 1
871 1120 1 FORMAL PARAMETERS:
872 1121 1
873 1122 1     LINE_NO.rw.r    Address of line number.
874 1123 1
875 1124 1     COL_NO.rw.r     Address of column number.
876 1125 1
877 1126 1 IMPLICIT INPUTS:
878 1127 1
879 1128 1     NONE
880 1129 1
881 1130 1 IMPLICIT OUTPUTS:
882 1131 1
883 1132 1     NONE
884 1133 1
885 1134 1 COMPLETION STATUS:
886 1135 1
887 1136 1     $$$_NORMAL      Normal successful completion
888 1137 1
889 1138 1 SIDE EFFECTS:
890 1139 1
891 1140 1     NONE
892 1141 1 --
893 1142 1
894 1143 2 BEGIN
895 1144 2
896 1145 2 BUILTIN
897 1146 2     ACTUALCOUNT,
898 1147 2     ACTUALPARAMETER ;
899 1148 2
900 1149 2 IF ACTUALCOUNT() EQL 2
901 1150 2 THEN
902 1151 3     RETURN ( SCR$SET_CURSOR (
903 1152 3         IF ACTUALPARAMETER(1) NEQ 0
904 1153 3         THEN
905 1154 3             IF .LINE_NO[0] EQL 0
906 1155 3             THEN
907 1156 4                 RETURN (LIB$_INVSCRPOS)
908 1157 3             ELSE
909 1158 3                 .LINE_NO[0]
910 1159 3             ELSE
911 1160 3                 RETURN ( LIB$_WRONUMARG),
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

LIB\$SET_CURSOR - Set Cursor to Character Positi

D 1

16-Sep-1984 02:28:18

14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742

[VMSLIB.SRC]SCREEN.B32;1

Page 25
(10)

LIE
V04

```
: 912      1161      3
: 913      1162      3
: 914      1163      3
: 915      1164      3
: 916      1165      3
: 917      1166      4
: 918      1167      3
: 919      1168      3
: 920      1169      3
: 921      1170      3
: 922      1171      3
: 923      1172      3
: 924      1173      2
: 925      1174      2
: 926      1175      2
: 927      1176      1

      IF ACTUALPARAMETER(2) NEQ 0
      THEN
        IF .COL_NO[0] EQL 0
        THEN
          RETURN (LIB$_INVSCRPOS)
        ELSE
          .COL_NO[0]
        ELSE
          RETURN ( LIB$_WRONUMARG ) )
      ELSE
        RETURN (LIB$_WRONUMARG ) ;
      END;
```

! End of routine LIB\$SET_CURSOR

			0000 00000	.ENTRY	LIB\$SET_CURSOR, Save nothing	: 1105
02		6C	91 00002	CMPB	(AP), #2	: 1149
		2E	12 00005	BNEQ	3\$	
	08	AC	D5 00007	TSTL	8(AP)	: 1162
		29	13 0000A	BEQL	3\$	
	08	BC	B5 0000C	TSTW	@COL_NO	: 1164
		10	13 0000F	BEQL	1\$	
50	08	BC	3C 00011	MOVZWL	@COL_NO, R0	: 1168
		50	DD 00015	PUSHL	R0	: 1164
	04	AC	D5 00017	TSTL	4(AP)	: 1152
		19	13 0001A	BEQL	3\$	
	04	BC	B5 0001C	TSTW	@LINE_NO	: 1154
		08	12 0001F	BNEQ	2\$	
50	00000000G	8F	D0 00021 1\$:	MOVL	#LIB\$_INVSCRPOS, R0	: 1156
			04 00028	RET		
50	04	BC	3C 00029 2\$:	MOVZWL	@LINE_NO, R0	: 1158
		50	DD 0002D	PUSHL	R0	: 1154
0000V	CF	02	FB 0002F	CALLS	#2, SCR\$SET_CURSOR	: 1152
			04 00034	RET		: 1174
50	00000000G	8F	D0 00035 3\$:	MOVL	#LIB\$_WRONUMARG, R0	
			04 0003C	RET		: 1176

; Routine Size: 61 bytes, Routine Base: _LIB\$CODE + 01AF

; 928 1177 1 !<BLF/PAGE>

```

930 1178 1 %SBTTL 'LIB$SET_SCROLL - Set Scrolling Region'
931 1179 1 GLOBAL ROUTINE [LIB$SET_SCROLL] (
932 1180 1     START_LINE : REF VECTOR [,WORD],
933 1181 1     END_LINE   : REF VECTOR [,WORD]
934 1182 1 ) =
935 1183 1
936 1184 1 ++
937 1185 1 FUNCTIONAL DESCRIPTION:
938 1186 1     This routine establishes a scrolling by setting the internal
939 1187 1     scrolling region parameters. Issues the escape sequence that
940 1188 1     establishes the region and preserves the cursor position.
941 1189 1
942 1190 1 CALLING SEQUENCE:
943 1191 1
944 1192 1     ret_status.wlc.v = LIB$SET_SCROLL (START_LINE.rw.r,
945 1193 1                                     END_LINE.rw.r)
946 1194 1
947 1195 1 FORMAL PARAMETERS:
948 1196 1
949 1197 1     START_LINE.rw.r      Address of first line of scrolling
950 1198 1                        region. Optional.
951 1199 1
952 1200 1     END_LINE.rw.r        Address of last line of scrolling
953 1201 1                        region. Optional.
954 1202 1
955 1203 1 IMPLICIT INPUTS:
956 1204 1
957 1205 1     NONE
958 1206 1
959 1207 1 IMPLICIT OUTPUTS:
960 1208 1
961 1209 1     NONE
962 1210 1
963 1211 1 COMPLETION STATUS:
964 1212 1
965 1213 1     $$$_NORMAL          Normal successful completion
966 1214 1
967 1215 1 SIDE EFFECTS:
968 1216 1
969 1217 1     NONE
970 1218 1 --
971 1219 1
972 1220 2 BEGIN
973 1221 2
974 1222 2 BUILTIN
975 1223 2     ACTUALCOUNT,
976 1224 2     ACTUALPARAMETER,
977 1225 2     AP,
978 1226 2     CALLG;
979 1227 2
980 1228 2 ++
981 1229 2 If no arguments, just call SCR$SET_SCROLL with original call list.
982 1230 2 ++
983 1231 2 IF ACTUALCOUNT() EQL 0
984 1232 2 THEN
985 1233 2     RETURN ( CALLG (.AP, SCR$SET_SCROLL));
986 1234 2
```

```

987 1235 2 !+
988 1236 2 Just 1 argument is a no-no. If LINE_NO is supplied, COL_NO must be
989 1237 2 supplied as well. Complain.
990 1238 2 -
991 1239 2 IF ACTUALCOUNT() EQL 1
992 1240 2 THEN
993 1241 2 IF ACTUALPARAMETER (1) EQL 0
994 1242 2 THEN ! 1 null arg - treat as no args
995 1243 3 RETURN ( SCR$SET_SCROLL ( ))
996 1244 2 ELSE ! 1 nonzero arg - error
997 1245 2 RETURN ( LIB$_WRONUMARG ) ;
998 1246 2
999 1247 2 !+
1000 1248 2 Check supplied arguments. If they are not specified (arglist entry = 0)
1001 1249 2 then pass on a zero. If they are specified, check for a value of zero
1002 1250 2 and complain. Call SCR$SET_SCROLL with appropriate combination of
1003 1251 2 arguments promoted to by-value.
1004 1252 2 -
1005 1253 2 IF ACTUALCOUNT() EQL 2
1006 1254 2 THEN
1007 1255 3 RETURN (SCR$SET_SCROLL (
1008 1256 3 IF ACTUALPARAMETER(1) NEQ 0
1009 1257 3 THEN
1010 1258 3 IF .START_LINE[0] EQL 0
1011 1259 3 THEN
1012 1260 4 RETURN (LIB$_INVSCRPOS)
1013 1261 3 ELSE
1014 1262 3 .START_LINE[0]
1015 1263 3 ELSE 0,
1016 1264 3 IF ACTUALPARAMETER(2) NEQ 0
1017 1265 3 THEN
1018 1266 3 IF .END_LINE[0] EQL 0
1019 1267 3 THEN
1020 1268 4 RETURN (LIB$_INVSCRPOS)
1021 1269 3 ELSE .END_LINE[0]
1022 1270 3 ELSE 0))
1023 1271 3
1024 1272 2 ELSE
1025 1273 2 RETURN (LIB$_WRONUMARG);
1026 1274 2
1027 1275 1 END;
```

! End of routine LIB\$SET_SCROLL

52	0000V	CF	9E	00002	.ENTRY	LIB\$SET_SCROLL, Save R2	1179
		6C	95	00007	MOVAB	SCR\$SET_SCROLL, R2	1231
		04	12	00009	TSTB	(AP)	1233
62		6C	FA	0000B	BNEQ	1\$	1239
			04	0000E	CALLG	(AP), SCR\$SET_SCROLL	1241
01		6C	91	0000F	RET		1243
		09	12	00012	CMPB	(AP), #1	
	04	AC	D5	00014	BNEQ	2\$	
		3D	12	00017	TSTL	4(AP)	
62		00	FB	00019	BNEQ	9\$	
					CALLS	#0, SCR\$SET_SCROLL	

LIB\$SCREEN
VC4-000

LIB\$SCREEN - Screen Management
LIB\$SET_SCROLL - Set Scrolling Region

G 1
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 28
(11)

02	6C	04	0001C	RET		1245
	34	91	0001D	CMPB	(AP), #2	1253
	08	12	00020	BNEQ	9\$	
	AC	D5	00022	TSTL	8(AP)	1265
	0D	13	00025	BEQL	3\$	
	08	B5	00027	TSTW	@END_LINE	1267
	14	13	0002A	BEQL	5\$	
50	08	BC	3C 0002C	MOVZWL	@END_LINE, R0	1270
	50	DD	00030	PUSHL	R0	1267
	02	11	00032	BRB	4\$	
	7E	D4	00034	CLRL	-(SP)	1265
	04	AC	D5 00036	TSTL	4(AP)	1256
	15	13	00039	BEQL	7\$	
	04	BC	B5 0003B	TSTW	@START_LINE	1258
	08	12	0003E	BNEQ	6\$	
50 00000000G	8F	D0	00040	MOVL	#LIB\$_INVSCRPOS, R0	1260
		04	00047	RET		
50	04	BC	3C 00048	MOVZWL	@START_LINE, R0	1262
	50	DD	0004C	PUSHL	R0	1258
	02	11	0004F	BRB	8\$	
	7E	D4	00050	CLRL	-(SP)	1256
62	02	FB	00052	CALLS	#2, SCR\$SET_SCROLL	
		04	00055	RET		1273
50 00000000G	8F	D0	0005D	MOVL	#LIB\$_WRONUMARG, R0	
		04	0005D	RET		1275

; Routine Size: 94 bytes, Routine Base: _LIB\$CODE + 01EC

; 1028 1276 1 !<BLF/PAGE>

```
1030 1277 1 *SBTTL 'LIB$UP_SCROLL - Up Scroll, Move Cursor Down One Line'
1031 1278 1 GLOBAL ROUTINE LIB$UP_SCROLL =
1032 1279 1
1033 1280 1 ++
1034 1281 1 FUNCTIONAL DESCRIPTION:
1035 1282 1
1036 1283 1 This routine causes the cursor to be moved down 1 line to the
1037 1284 1 same column of the following line. If the cursor was on the
1038 1285 1 bottom line to begin with, it stays where it was, but all the
1039 1286 1 information on the screen appears to move up one line. The
1040 1287 1 information that was on the top line is lost and a blank line
1041 1288 1 appears at the bottom.
1042 1289 1
1043 1290 1 If a scrolling region is active ( on a VT100 or in mapping)
1044 1291 1 then the logic above applies to the top and bottom lines of
1045 1292 1 the scrolling region rather than the top and bottom line of
1046 1293 1 the screen. If an UP_SCROLL is performed on the bottom line of
1047 1294 1 the screen, or a DOWN_SCROLL is performed on the top line of the
1048 1295 1 screen, while a scrolling region is active then no screen
1049 1296 1 movement takes place unless the top or bottom line of the screen
1050 1297 1 corresponds to the top or bottom line of the scrolling region.
1051 1298 1
1052 1299 1 CALLING SEQUENCE:
1053 1300 1
1054 1301 1 ret_status.wlc.v = LIB$UP_SCROLL ()
1055 1302 1
1056 1303 1 FORMAL PARAMETERS:
1057 1304 1
1058 1305 1 NONE
1059 1306 1
1060 1307 1 IMPLICIT INPUTS:
1061 1308 1
1062 1309 1 NONE
1063 1310 1
1064 1311 1 IMPLICIT OUTPUTS:
1065 1312 1
1066 1313 1 NONE
1067 1314 1
1068 1315 1 COMPLETION STATUS:
1069 1316 1
1070 1317 1 $$$_NORMAL Normal successful completion
1071 1318 1
1072 1319 1 SIDE EFFECTS:
1073 1320 1
1074 1321 1 NONE
1075 1322 1 --
1076 1323 1
1077 1324 1 +
1078 1325 1 Equate to SCR$ entry point.
1079 1326 1 -
1080 1327 1
1081 1328 1 GLOBAL BIND ROUTINE LIB$UP_SCROLL = SCR$UP_SCROLL ;
1082 1329 1 !<BLF/PAGE>
```

```
: 1084 1330 1 %SBTTL 'SCR$DOWN_SCROLL - Down Scroll, Move Cursor up One Line'
: 1085 1331 1 GLOBAL ROUTINE SCR$DOWN_SCROLL =
: 1086 1332 1 ++
: 1087 1333 1 FUNCTIONAL DESCRIPTION:
: 1088 1334 1
: 1089 1335 1 This routine causes the cursor to be moved up 1 line to the
: 1090 1336 1 same column of the previous line. If the cursor was on the
: 1091 1337 1 top line to begin with, it stays where it was, but all the
: 1092 1338 1 information on the screen appears to move down one line. The
: 1093 1339 1 information that was on the bottom line is lost and a blank line
: 1094 1340 1 appears at the top.
: 1095 1341 1
: 1096 1342 1 If a scrolling region is active ( on a VT100 or in mapping)
: 1097 1343 1 then the logic above applies to the top and bottom lines of
: 1098 1344 1 the scrolling region rather than the top and bottom line of
: 1099 1345 1 the screen. If an UP_SCROLL is performed on the bottom line of
: 1100 1346 1 the screen, or a DOWN_SCROLL is performed on the top line of the
: 1101 1347 1 screen, while a scrolling region is active then no screen
: 1102 1348 1 movement takes place unless the top or bottom line of the screen
: 1103 1349 1 corresponds to the top or bottom line of the scrolling region.
: 1104 1350 1
: 1105 1351 1 CALLING SEQUENCE:
: 1106 1352 1
: 1107 1353 1 ret_status.wlc.v = SCR$DOWN_SCROLL ()
: 1108 1354 1
: 1109 1355 1 FORMAL PARAMETERS:
: 1110 1356 1
: 1111 1357 1 NONE
: 1112 1358 1
: 1113 1359 1 IMPLICIT INPUTS:
: 1114 1360 1
: 1115 1361 1 NONE
: 1116 1362 1
: 1117 1363 1 IMPLICIT OUTPUTS:
: 1118 1364 1
: 1119 1365 1 NONE
: 1120 1366 1
: 1121 1367 1 COMPLETION STATUS:
: 1122 1368 1
: 1123 1369 1 SSS_NORMAL Normal successful completion
: 1124 1370 1
: 1125 1371 1 SIDE EFFECTS:
: 1126 1372 1
: 1127 1373 1 NONE
: 1128 1374 1 --
: 1129 1375 1
: 1130 1376 2 BEGIN
: 1131 1377 2
: 1132 1378 2 MACRO
: 1133 1379 2 VT05_DOWN = %STRING (%CHAR(VT05_CUP), %CHAR(NULL), %CHAR(NULL), %CHAR(NULL), %CHAR(NULL))%,
: 1134 1380 2 VT52_DOWN = %STRING (%CHAR(ESC), %CHAR(VT52_DWN))%,
: 1135 1381 2 VT100_DOWN = %STRING (%CHAR(ESC), %CHAR(VT100_DWN))%;
: 1136 1382 2
: 1137 1383 2 LOCAL
: 1138 1384 2 STATUS, ! used for calls
: 1139 1385 2 TCB, ! terminal control block
: 1140 1386 2 DOWN_BUF : BLOCK [8,BYTE], ! buffer dsc for OUTPUT
```



```
1141 1387 2      TERM_TYPE;                ! terminal type
1142 1388
1143 1389
1144 1390 2      MAP
1145 1391 2          TCB : REF BLOCK [,BYTE];
1146 1392 2      STATUS = SCR$$GET_TYPE_R3 (SCR$C_DOWN_SCROLL; TCB, TERM_TYPE);
1147 1393 2          ! get current control block,
1148 1394 2      IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTFOREIGN
1149 1395 2      THEN RETURN .STATUS;
1150 1396
1151 1397 2      TCB [SCR$L_LINE] = .TCB [SCR$L_LINE] - 1;
1152 1398 2      IF .TCB [SCR$L_LINE] LEQ 0
1153 1399 2      THEN
1154 1400 2          TCB [SCR$L_LINE] = 1;                ! limit cursor to line 1
1155 1401 2
1156 1402 2      IF .TCB [SCR$L_CHARMAP] NEQ 0 AND
1157 1403 2          .TCB [SCR$L_BUFFER] NEQ 0
1158 1404 2      THEN
1159 1405 2          RETURN (SS$_NORMAL);                ! do nothing if mapping active and
1160 1406 2          ! buffering enabled
1161 1407
1162 1408 2      DOWN_BUF [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1163 1409 2      DOWN_BUF [DSC$B_CLASS] = DSC$K_CLASS_S;
1164 1410
1165 1411 2      CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
1166 1412 2          SET
1167 1413 2              [UNKNOWN] :
1168 1414 2                  BEGIN
1169 1415 2                      DOWN_BUF [DSC$W_LENGTH] = 0;
1170 1416 2                      DOWN_BUF [DSC$A_POINTER] = 0;
1171 1417 2                  END;
1172 1418
1173 1419 2              [VT05] :
1174 1420 2                  BEGIN
1175 1421 2                      DOWN_BUF [DSC$W_LENGTH] = %CHARCOUNT (VT05_DOWN);
1176 1422 2                      DOWN_BUF [DSC$A_POINTER] = UPLIT (BYTE (VT05_DOWN));
1177 1423 2                  END;
1178 1424
1179 1425 2              [VT52] :
1180 1426 2                  BEGIN
1181 1427 2                      DOWN_BUF [DSC$W_LENGTH] = %CHARCOUNT (VT52_DOWN);
1182 1428 2                      DOWN_BUF [DSC$A_POINTER] = UPLIT (BYTE (VT52_DOWN));
1183 1429 2                  END;
1184 1430
1185 1431 2              [VT100] :
1186 1432 2                  BEGIN
1187 1433 2                      DOWN_BUF [DSC$W_LENGTH] = %CHARCOUNT (VT100_DOWN);
1188 1434 2                      DOWN_BUF [DSC$A_POINTER] = UPLIT (BYTE (VT100_DOWN));
1189 1435 2                  END;
1190 1436
1191 1437 2      TES;
1192 1438
1193 1439 2      RETURN (STATUS = OUTPUT (.TCB, DOWN_BUF));
1194 1440 2          ! output sequence for this terminal type
1195 1441 2
1196 1442 1      END;                ! End of routine SCR$DOWN_SCROLL
```

```

00 00 00 00 1A 0024A .BLKB 2
00 00 00 00 1A 0024C P.AAA: .ASCII <26><0><0><0><0>
00 00 00 00 1A 00251 .BLKB 3
00 00 00 00 49 1B 00254 P.AAB: .ASCII <27>\I\
00 00 00 00 49 1B 00256 .BLKB 2
00 00 00 00 4D 1B 00258 P.AAC: .ASCII <27>\M\

```

				000C	00000	.ENTRY	SCR\$DOWN_SCROLL, Save R2,R3	: 1331
	5E			08	C2	SUBL2	#8, SP	:
	50			05	D0	MOVL	#5, R0	: 1392
				0000V	30	B\$BW	SCR\$\$GET_TYPE_R3	:
	5A			50	E9	BLBC	STATUS, 9\$	: 1394
	04		0A	A1	91	CMPB	10(TCB), #4	:
				54	13	BEQL	9\$	:
	04		24	A1	F5	SOBGTR	36(TCB), 1\$	: 1397
	24	A1		01	D0	MOVL	#1, 36(TCB)	: 1400
			18	A1	D5	TSTL	24(TCB)	: 1402
				09	13	BEQL	2\$	:
			04	A1	D5	TSTL	4(TCB)	: 1403
				04	13	BEQL	2\$	:
	50			01	D0	MOVL	#1, R0	: 1405
				04	00029	RET		:
	02	AE	010E	8F	B0	MOVW	#270, DOWN_BUF+2	: 1408
		00		52	CF	CASEL	TERM TYPE, #0, #3	: 1411
0023		03 0019		000F	0008	.WORD	4\$-3\$, - 5\$-3\$, - 6\$-3\$, - 7\$-3\$	:
				6E	B4	CLRW	DOWN_BUF	: 1415
			04	AE	D4	CLRL	DOWN_BUF+4	: 1416
				1C	11	BRB	8\$	: 1411
				05	B0	MOVW	#5, DOWN_BUF	: 1421
	04	6E		AF	9E	MOVAB	P.AAA, DOWN_BUF+4	: 1422
		AE	A9	12	11	BRB	8\$	: 1411
				02	B0	MOVW	#2, DOWN_BUF	: 1427
	04	6E		AF	9E	MOVAB	P.AAB, DOWN_BUF+4	: 1428
		AE	A7	08	11	BRB	8\$	: 1411
				02	B0	MOVW	#2, DOWN_BUF	: 1433
	04	6E		AF	9E	MOVAB	P.AAC, DOWN_BUF+4	: 1434
		AE	A1	8F	BB	PUSHR	#*M<R1, SP>	: 1439
			4002	02	FB	CALLS	#2, OUTPUT	: 1442
	0000V	CF		04	00068	RET		: 1442

```
; Routine Size: 105 bytes,    Routine Base: _LIB$CODE + 025A
```

; 1197 1443 1 !<BLF/PAGE>

1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237

```

1444 1 *SBTTL 'SCR$ERASE - Obsolete version of SCR$ERASE_PAGE'
1445 1   GLOBAL ROUTINE SCR$ERASE =
1446 1
1447 1 ++
1448 1   FUNCTIONAL DESCRIPTION:
1449 1
1450 1       This routine is obsolete and should not be called.  It is still
1451 1       defined because its entry cannot be removed from SCRVECTOR.
1452 1
1453 1   CALLING SEQUENCE:
1454 1
1455 1
1456 1   FORMAL PARAMETERS:
1457 1
1458 1       NONE
1459 1
1460 1   IMPLICIT INPUTS:
1461 1
1462 1       NONE
1463 1
1464 1   IMPLICIT OUTPUTS:
1465 1
1466 1       NONE
1467 1
1468 1   COMPLETION STATUS:
1469 1
1470 1       SS$_NORMAL      Normal successful completion
1471 1
1472 1   SIDE EFFECTS:
1473 1
1474 1       NONE
1475 1   --
1476 1
1477 1   +
1478 1   Equate to current entry point.
1479 1   -
1480 1
1481 1 GLOBAL BIND ROUTINE SCR$ERASE = SCR$ERASE_PAGE ;
1482 1 !<BLF/PAGE>

```

LIB
V04

[illegible]

```
: 1239 1483 1 %SBTTL 'SCR$ERASE_LINE - Erase Line'
: 1240 1484 1 GLOBAL ROUTINE SCR$ERASE_LINE (
: 1241 1485 1 LINE_NO : WORD,
: 1242 1486 1 COL_NO : WORD
: 1243 1487 1 ) =
: 1244 1488 1 ++
: 1245 1489 1 FUNCTIONAL DESCRIPTION:
: 1246 1490 1
: 1247 1491 1 This routine causes the screen to be erased from the specified
: 1248 1492 1 position to the end of the line. If the position is not
: 1249 1493 1 specified, the current screen position is assumed.
: 1250 1494 1
: 1251 1495 1 CALLING SEQUENCE:
: 1252 1496 1
: 1253 1497 1 ret_status.wlc.v = SCR$ERASE_LINE ([LINE_NO.rw.v,
: 1254 1498 1 COL_NO.rw.v])
: 1255 1499 1
: 1256 1500 1 FORMAL PARAMETERS:
: 1257 1501 1
: 1258 1502 1 LINE_NO.rw.v Optional. Line number where erasing starts.
: 1259 1503 1 If this argument is omitted, COL_NO is ignored
: 1260 1504 1 as well, and current position will be used.
: 1261 1505 1
: 1262 1506 1 COL_NO.rw.v Optional. Column number where erasing starts.
: 1263 1507 1 If this argument is omitted, LINE_NO is ignored
: 1264 1508 1 as well, and current position will be used.
: 1265 1509 1
: 1266 1510 1 IMPLICIT INPUTS:
: 1267 1511 1
: 1268 1512 1 NONE
: 1269 1513 1
: 1270 1514 1 IMPLICIT OUTPUTS:
: 1271 1515 1
: 1272 1516 1 NONE
: 1273 1517 1
: 1274 1518 1 COMPLETION STATUS:
: 1275 1519 1
: 1276 1520 1 SS$_NORMAL Normal successful completion
: 1277 1521 1
: 1278 1522 1 SIDE EFFECTS:
: 1279 1523 1
: 1280 1524 1 NONE
: 1281 1525 1 --
: 1282 1526 1
: 1283 1527 2 BEGIN
: 1284 1528 2
: 1285 1529 2 BUILTIN
: 1286 1530 2 ACTUAL COUNT,
: 1287 1531 2 ACTUAL PARAMETER ;
: 1288 1532 2
: 1289 1533 2 MACRO
: 1290 1534 2 VT05_LINE = %STRING (%CHAR(VT05_EOL), %CHAR(NULL), %CHAR(NULL))% ! VT05 (w/fill)
: 1291 1535 2 VT52_LINE = %STRING (%CHAR(ESC), %CHAR(VT52_EOL))% ! VT52
: 1292 1536 2 VT100_LINE = %STRING (%CHAR(ESC), %CHAR(LB), %CHAR(VT100_EOL))% ! VT100
: 1293 1537 2
: 1294 1538 2 LITERAL
: 1295 1539 2 K_CURSOR_ARGS = 2, ! 2 args if line_no & col_no present
```

```
1296 1540 2      K_LINE_SIZ = 25;      ! size for erase line seq buffer
1297 1541 2      ! ***** known sequences will not
1298 1542 2      ! ***** exceed this size. in the
1299 1543 2      ! ***** future it may be necessary
1300 1544 2      ! ***** to increase this.
1301 1545 2
1302 1546 2      LOCAL
1303 1547 2      TERM_TYPE,      ! terminal type
1304 1548 2      TCB,      ! ptr to terminal control block
1305 1549 2      STATUS;      ! status ret'd by called routines
1306 1550 2      MAP
1307 1551 2      TCB : REF BLOCK [,BYTE];
1308 1552 2
1309 1553 2      STATUS = SCR$$GET_TYPE_A3 (SCR$C_ERASE_LINE; TCB, TERM_TYPE);
1310 1554 2      ! get current TCB
1311 1555 2      IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VIFOREIGN
1312 1556 2      THEN RETURN (.STATUS);
1313 1557 2
1314 1558 2      !+
1315 1559 2      ! If mapping is active, just clear the character and attribute maps.
1316 1560 2      ! If mapping is inactive, set the cursor (if line_no and col_no
1317 1561 2      ! were specified) and output the erase line sequence.
1318 1562 2      !-
1319 1563 2
1320 1564 2      IF .TCB [SCR$L_CHARMAP] NEQ 0      ! mapping active
1321 1565 2      THEN
1322 1566 3      BEGIN
1323 1567 3      IF ACTUALCOUNT () EQL K_CURSOR_ARGS
1324 1568 3      THEN
1325 1569 4      BEGIN      ! reset cursor pos if line, col spec
1326 1570 4      TCB [SCR$L_LINE] = .LINE_NO;
1327 1571 4      TCB [SCR$L_COLUMN] = .COL_NO;
1328 1572 3      END;
1329 1573 3      FILL_MAP (.TCB,
1330 1574 3      .TCB [SCR$L_LINE],
1331 1575 3      .TCB [SCR$L_COLUMN],
1332 1576 3      .TCB [SCR$L_LINE],
1333 1577 3      .TCB [SCR$W_DEVWIDTH],
1334 1578 3      ' ');      ! fill maps with spaces
1335 1579 3      TCB [SCR$L_ATTRMASK] = 0;      ! clear OR'd attr mask
1336 1580 3      RETURN (SS$NORMAL);
1337 1581 3      END
1338 1582 2      ELSE      ! mapping inactive
1339 1583 2      BEGIN
1340 1584 3      LOCAL
1341 1585 3      BUFFER : BLOCK [8, BYTE],      ! dsc for buffer
1342 1586 3      CUR_BUF_SIZ : INITIAL (0),      ! size of buffer
1343 1587 3      OUT_STATUS,      ! status ret'd by OUTPUT
1344 1588 3      ERASE_LEN,      ! length of erase sequence
1345 1589 3      ERASE_PTR,      ! ptr to erase sequence
1346 1590 3      TEMP_BUF : VECTOR [K_LINE_SIZ, BYTE];      ! buffer to hold set cursor and
1347 1591 3      ! erase page sequences
1348 1592 3
1349 1593 3      BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1350 1594 3      BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
1351 1595 3      BUFFER [DSC$A_POINTER] = TEMP_BUF;
1352 1596 3      !+
```

```
1353 1597 3 | Set cursor only if line no and col no were specified.
1354 1598 3 | Otherwise just default to current cursor position.
1355 1599 3 |
1356 1600 3 IF ACTUALCOUNT () EQL K_CURSOR_ARGS
1357 1601 3 THEN
1358 1602 4 BEGIN
1359 1603 4 STATUS = SET_CURSOR (.TCB, .LINE NO, .COL NO,
1360 1604 4 .BUFFER [DSC$A_POINTER], CUR_BUF_SIZ);
1361 1605 4 IF NOT .STATUS THEN RETURN (.STATUS);
1362 1606 3 END;
1363 1607 3
1364 1608 3 CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
1365 1609 3 SET
1366 1610 3
1367 1611 3 [UNKNOWN]:
1368 1612 4 BEGIN
1369 1613 4 ERASE_LEN = 0;
1370 1614 4 ERASE_PTR = 0;
1371 1615 3 END;
1372 1616 3
1373 1617 3 [VT05]:
1374 1618 4 BEGIN
1375 1619 4 ERASE_LEN = %CHARCOUNT (VT05_LINE);
1376 1620 4 ERASE_PTR = UPLIT ( BYTE (VT05_LINE));
1377 1621 3 END;
1378 1622 3
1379 1623 3 [VT52]:
1380 1624 4 BEGIN
1381 1625 4 ERASE_LEN = %CHARCOUNT (VT52_LINE);
1382 1626 4 ERASE_PTR = UPLIT (BYTE (VT52_LINE));
1383 1627 3 END;
1384 1628 3
1385 1629 3 [VT100]:
1386 1630 4 BEGIN
1387 1631 4 ERASE_LEN = %CHARCOUNT (VT100_LINE);
1388 1632 4 ERASE_PTR = UPLIT (BYTE (VT100_LINE));
1389 1633 3 END;
1390 1634 3
1391 1635 3 TES;
1392 1636 3
1393 1637 3 CH$MOVE (.ERASE_LEN,
1394 1638 3 .ERASE_PTR, .BUFFER [DSC$A_POINTER] + .CUR_BUF_SIZ);
1395 1639 3 ! append erase seq to set cursor seq
1396 1640 3 BUFFER [DSC$W_LENGTH] = .ERASE_LEN + .CUR_BUF_SIZ;
1397 1641 3 ! update length of buffer
1398 1642 3 OUT_STATUS = OUTPUT (.TCB, BUFFER);
1399 1643 3
1400 1644 3 IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
1401 1645 3
1402 1646 3 RETURN (SS$_NORMAL);
1403 1647 3
1404 1648 2 END;
1405 1649 2
1406 1650 1 END; ! End of routine SCR$ERASE_LINE
```

```

00  00  1E  002C3      .BLKB      1
      002C4 P.AAD:  .ASCII    <30><0><0>
      002C7      .BLKB      1
      4B  1B  002C8 P.AAE:  .ASCII    <27>\K\
      002CA      .BLKB      2
      4B  5B  1B  002CC P.AAF:  .ASCII    <27>\[K\

```

	5E		28	00FC	00000	.ENTRY	SCR\$ERASE_LINE, Save R2,R3,R4,R5,R6,R7	: 1484
	50		03	D0	00002	SUBL2	#40, SP	:
			03	D0	00005	MOVL	#3, R0	: 1553
	53		0000V	30	00008	BSBW	SCR\$\$GET_TYPE_R3	:
	56		50	D0	0000B	MOVL	R0, STATUS	:
	5F		51	D0	0000E	MOVL	R1, R6	:
	04		53	E9	00011	BLBC	STATUS, 3\$	: 1555
		0A	A6	91	00014	CMPB	10(TCB), #4	:
			59	13	00018	BEQL	3\$	:
		18	A6	D5	0001A	TSTL	24(TCB)	: 1564
			28	13	0001D	BEQL	2\$	:
	02		6C	91	0001F	CMPB	(AP), #2	: 1567
			0A	12	00022	BNEQ	1\$	:
24	A6	04	AC	3C	00024	MOVZWL	LINE_NO, 36(TCB)	: 1570
28	A6	08	AC	3C	00029	MOVZWL	COL_NO, 40(TCB)	: 1571
			20	DD	0002E	PUSHL	#32	: 1573
	7E	0C	A6	3C	00030	MOVZWL	12(TCB), -(SP)	: 1577
		24	A6	DD	00034	PUSHL	36(TCB)	: 1576
	7E	24	A6	7D	00037	MOVQ	36(TCB), -(SP)	: 1574
			56	DD	0003B	PUSHL	TCB	: 1573
0000V	CF		06	FB	0003D	CALLS	#6, FILL_MAP	:
		2C	A6	D4	00042	CLRL	44(TCB)	: 1579
			79	11	00045	BRB	11\$	: 1583
			6E	D4	00047	CLRL	CUR_BUF_SIZ	:
22	AE	010E	8F	B0	00049	MOVW	#270, BUFFER+2	: 1593
24	AE	04	AE	9E	0004F	MOVAB	TEMP_BUF, BUFFER+4	: 1595
	02		6C	91	00054	CMPB	(AP), #2	: 1600
			1E	12	00057	BNEQ	4\$	:
			5E	DD	00059	PUSHL	SP	: 1603
		28	AE	DD	0005B	PUSHL	BUFFER+4	: 1604
	7E	08	AC	3C	0005E	MOVZWL	COL_NO, -(SP)	: 1603
	7E	04	AC	3C	00062	MOVZWL	LINE_NO, -(SP)	:
			56	DD	00066	PUSHL	TCB	:
0000V	CF		05	FB	00068	CALLS	#5, SET_CURSOR	:
	53		50	D0	0006D	MOVL	R0, STATUS	:
	04		53	E8	00070	BLBS	STATUS, 4\$	: 1605
	50		53	D0	00073	MOVL	STATUS, R0	:
			04	00076	RET			:
	00		52	CF	00077	CASEL	TERM_TYPE, #0, #3	: 1608
000E			0008	0007B	5\$: .WORD	6\$-5\$, -		:
						7\$-5\$, -		:
						8\$-5\$, -		:
						9\$-5\$		:
			57	D4	00083	CLRL	ERASE_LEN	: 1613
			50	D4	00085	CLRL	ERASE_PTR	: 1614
			1C	11	00087	BRB	10\$	: 1608
57			03	D0	00089	MOVL	#3, ERASE_LEN	: 1619
50	FF65		CF	9E	0008C	MOVAB	P.AAD, ERASE_PTR	: 1620

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SCR\$ERASE_LINE - Erase Line

D 2
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 38
(15)

LIB
V04

				2	11	00091	BRB	10\$	:	1608	:	1	
		57		02	D0	00093	8\$:	MOVL	#2, ERASE_LEN	:	1625	:	1
		50	FF5F	CF	9E	00096		MOVAB	P.AAE, ERASE_PTR	:	1626	:	1
				08	11	00098		BRB	10\$	:	1608	:	1
		57		03	D0	0009D	9\$:	MOVL	#3, ERASE_LEN	:	1631	:	1
		50	FF59	CF	9E	000A0		MOVAB	P.AAF, ERASE_PTR	:	1632	:	1
	51	24	AE	6E	C1	000A5	10\$:	ADDL3	CUR_BUF_SIZ, -BUFFER+4, R1	:	1638	:	1
	61		60	57	28	000AA		MOVCL	ERASE_LEN, (ERASE_PTR), (R1)	:		:	1
20	AE		57	6E	A1	000AE		ADDW3	CUR_BUF_SIZ, ERASE_LEN, BUFFER	:	1640	:	1
				AE	9F	000B3		PUSHAB	BUFFER	:	1642	:	1
				56	DD	000B6		PUSHL	TCB	:		:	1
		0000V	CF	02	FB	000B8		CALLS	#2, OUTPUT	:		:	
			03	50	E9	000BD		BLBC	OUT_STATUS, 12\$	:	1644	:	
			50	01	D0	000C0	11\$:	MOVL	#1, -R0	:	1646	:	
				04	000C3	12\$:	RET			:	1650	:	

; Routine Size: 196 bytes, Routine Base: _LIB\$CODE + 02CF

; 1407 1651 1 !<BLF/PAGE>


```
1409 1652 1 %SBTTL 'SCR$ERASE_PAGE - Erase Page'
1410 1653 1 GLOBAL ROUTINE SCR$ERASE_PAGE (
1411 1654 1 LINE_NO : WORD,
1412 1655 1 COL_NO : WORD
1413 1656 1 ) =
1414 1657 1 ++
1415 1658 1 FUNCTIONAL DESCRIPTION:
1416 1659 1
1417 1660 1 This routine causes the screen to be erased from the specified
1418 1661 1 position to the end of the screen. If the position is not
1419 1662 1 specified, the current screen position is assumed.
1420 1663 1
1421 1664 1 CALLING SEQUENCE:
1422 1665 1
1423 1666 1 ret_status.wlc.v = SCR$ERASE_PAGE ([LINE_NO.rw.v,
1424 1667 1 COL_NO.rw.v])
1425 1668 1
1426 1669 1 FORMAL PARAMETERS:
1427 1670 1
1428 1671 1 LINE_NO.rw.v Optional. Line number at which to start
1429 1672 1 erasing. If omitted, COL_NO will be ignored
1430 1673 1 as well and current cursor position will
1431 1674 1 be used.
1432 1675 1
1433 1676 1 COL_NO.rw.v Optional. Column number at which to start
1434 1677 1 erasing. If omitted, LINE_NO will be ignored
1435 1678 1 as well and current cursor position will
1436 1679 1 be used.
1437 1680 1
1438 1681 1 IMPLICIT INPUTS:
1439 1682 1
1440 1683 1 NONE
1441 1684 1
1442 1685 1 IMPLICIT OUTPUTS:
1443 1686 1
1444 1687 1 NONE
1445 1688 1
1446 1689 1 COMPLETION STATUS:
1447 1690 1
1448 1691 1 $$$_NORMAL Normal successful completion
1449 1692 1
1450 1693 1 SIDE EFFECTS:
1451 1694 1
1452 1695 1 NONE
1453 1696 1 --
1454 1697 1
1455 1698 2 BEGIN
1456 1699 2
1457 1700 2 BUILTIN
1458 1701 2 ACTUALCOUNT,
1459 1702 2 ACTUALPARAMETER ;
1460 1703 2
1461 1704 2 MACRO
1462 1705 2 VT05_ERASE = %STRING ( %CHAR(VT05_EOS), %CHAR(NULL), %CHAR(NULL), %CHAR(NULL), %CHAR(NULL) )%, ! VT05 (
1463 1706 2 VT52_ERASE = %STRING ( %CHAR(ESC), %CHAR(VT52_EOS))%, ! VT52
1464 1707 2 VT100_ERASE = %STRING ( %CHAR(ESC), %CHAR(LB), %CHAR(VT100_EOS))%, ! VT100
1465 1708 2
```

: R

: 1

```
LIB$SCREEN
V04-000

: 1466 1709 2 LITERAL
: 1467 1710 2 K_CURSOR_ARGS = 2,
: 1468 1711 2 K_ERASE_SIZE = 25;
: 1469 1712 2
: 1470 1713 2
: 1471 1714 2
: 1472 1715 2
: 1473 1716 2
: 1474 1717 2
: 1475 1718 2
: 1476 1719 2
: 1477 1720 2
: 1478 1721 2
: 1479 1722 2
: 1480 1723 2
: 1481 1724 2
: 1482 1725 2
: 1483 1726 2
: 1484 1727 2
: 1485 1728 2
: 1486 1729 2
: 1487 1730 2
: 1488 1731 2
: 1489 1732 2
: 1490 1733 2
: 1491 1734 2
: 1492 1735 2
: 1493 1736 2
: 1494 1737 3
: 1495 1738 3
: 1496 1739 3
: 1497 1740 4
: 1498 1741 4
: 1499 1742 4
: 1500 1743 3
: 1501 1744 3
: 1502 1745 3
: 1503 1746 3
: 1504 1747 3
: 1505 1748 3
: 1506 1749 3
: 1507 1750 3
: 1508 1751 3
: 1509 1752 3
: 1510 1753 2
: 1511 1754 3
: 1512 1755 3
: 1513 1756 3
: 1514 1757 3
: 1515 1758 3
: 1516 1759 3
: 1517 1760 3
: 1518 1761 3
: 1519 1762 3
: 1520 1763 3
: 1521 1764 3
: 1522 1765 3

LITERAL
K_CURSOR_ARGS = 2,
K_ERASE_SIZE = 25;

LOCAL
TERM_TYPE,
TCB,
STATUS;

MAP
TCB : REF BLOCK [,BYTE];

STATUS = SCR$$GET_TYPE_R3 (SCR$C_ERASE_PAGE; TCB, TERM_TYPE);
! get current TCB
IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTForeign
THEN RETURN (.STATUS);

!+
! If mapping is active, just clear the character and attribute maps.
! If mapping is inactive, set the cursor (if line_no and col_no
! were specified) and output the erase page sequence.
!-

IF .TCB [SCR$L_CHARMAP] NEQ 0 ! mapping active
THEN
BEGIN
IF ACTUALCOUNT () EQL K_CURSOR_ARGS
THEN
BEGIN ! store new cursor pos if line, col spec
TCB [SCR$L_LINE] = .LINE_NO;
TCB [SCR$L_COLUMN] = .COL_NO;
END;
FILL_MAP (.TCB,
.TCB [SCR$L_LINE],
.TCB [SCR$L_COLUMN],
.TCB [SCR$W_DEVPAGSIZE],
.TCB [SCR$W_DEVWIDTH],
XC' '); ! fill maps with spaces
TCB [SCR$L_ATTRMASK] = 0; ! clear OR'd attr mask
RETURN (SS$NORMAL);
END
ELSE ! mapping inactive
BEGIN
LOCAL
BUFFER : BLOCK [8, BYTE], ! dsc for buffer
CUR_BUF_SIZE : INITIAL (0), ! size of buffer
OUT_STATUS, ! status retd by OUTPUT
ERASE_LEN, ! length of erase sequence
ERASE_PTR, ! ptr to erase sequence
TEMP_BUF : VECTOR [K_ERASE_SIZE, BYTE]; ! buffer to hold erase_page sequence

BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
BUFFER [DSC$A_POINTER] = TEMP_BUF;
```

```
1523 1766 3
1524 1767 3
1525 1768 3
1526 1769 3
1527 1770 3
1528 1771 3
1529 1772 3
1530 1773 4
1531 1774 4
1532 1775 4
1533 1776 4
1534 1777 3
1535 1778 3
1536 1779 3
1537 1780 3
1538 1781 3
1539 1782 4
1540 1783 4
1541 1784 4
1542 1785 3
1543 1786 3
1544 1787 3
1545 1788 4
1546 1789 4
1547 1790 4
1548 1791 3
1549 1792 3
1550 1793 3
1551 1794 4
1552 1795 4
1553 1796 4
1554 1797 3
1555 1798 3
1556 1799 3
1557 1800 4
1558 1801 4
1559 1802 4
1560 1803 3
1561 1804 3
1562 1805 3
1563 1806 3
1564 1807 3
1565 1808 3
1566 1809 3
1567 1810 3
1568 1811 3
1569 1812 3
1570 1813 3
1571 1814 3
1572 1815 3
1573 1816 3
1574 1817 2
1575 1818 2
1576 1819 1

+
Set cursor only if line no and col no were specified.
Otherwise just default to current cursor position.
IF ACTUALCOUNT () EQL K_CURSOR_ARGS
THEN
  BEGIN
    STATUS = SET_CURSOR (.TCB, .LINE_NO, .COL_NO,
      .BUFFER [DSC$A_POINTER], CUR_BUF_SIZ);
    IF NOT .STATUS THEN RETURN (.STATUS);
  END;
CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
  SET
    [UNKNOWN]:
      BEGIN
        ERASE_LEN = 0;
        ERASE_PTR = 0;
      END;
    [VT05]:
      BEGIN
        ERASE_LEN = %CHARCOUNT (VT05_ERASE);
        ERASE_PTR = UPLIT (BYTE (VT05_ERASE));
      END;
    [VT52]:
      BEGIN
        ERASE_LEN = %CHARCOUNT (VT52_ERASE);
        ERASE_PTR = UPLIT (BYTE (VT52_ERASE));
      END;
    [VT100]:
      BEGIN
        ERASE_LEN = %CHARCOUNT (VT100_ERASE);
        ERASE_PTR = UPLIT (BYTE (VT100_ERASE));
      END;
  TES;
CH$MOVE (.ERASE_LEN,
  .ERASE_PTR, .BUFFER [DSC$A_POINTER] + .CUR_BUF_SIZ);
BUFFER [DSC$W_LENGTH] = .ERASE_LEN + .CUR_BUF_SIZ;
OUT_STATUS = OUTPUT (.TCB, BUFFER);
IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
RETURN (SS$_NORMAL);
END;

! End of routine SCR$ERASE_PAGE
```

```
00 00 00 00 1F 00393 .BLKB 1
00394 P.AAG: .ASCII <31><0><0><0><0>
00399 .BLKB 3
4A 1B 0039C P.AAH: .ASCII <27>\J\
0039E .BLKB 2
4A 5B 1B 003A0 P.AAI: .ASCII <27>\[J\
```

```
00FC 00000 .ENTRY SCR$ERASE_PAGE, Save R2,R3,R4,R5,R6,R7 : 1653
5E 28 C2 00002 .SUBL2 #40, SP : 1724
50 02 D0 00005 .MOVL #2, R0
0000V 30 00008 .BSBW SCR$GET_TYPE_R3
53 50 D0 0000B .MOVL R0, STATUS
56 51 D0 0000E .MOVL R1, R6
60 53 E9 00011 .BLBC STATUS, 3$ : 1726
04 0A A6 91 00014 .CMPB 10(TCB), #4
5A 13 00018 .BEQL 3$
18 A6 D5 0001A .TSTL 24(TCB) : 1735
29 13 0001D .BEQL 2$
02 6C 91 0001F .CMPB (AP), #2 : 1738
0A 12 00022 .BNEQ 1$
24 A6 04 AC 3C 00024 .MOVZWL LINE_NO, 36(TCB) : 1741
28 A6 08 AC 3C 00029 .MOVZWL COL_NO, 40(TCB) : 1742
7E 0C A6 3C 00030 1$: .PUSHL #32- : 1744
7E 0E A6 3C 00034 .MOVZWL 12(TCB), -(SP) : 1748
7E 24 A6 7D 00038 .MOVZWL 14(TCB), -(SP) : 1747
56 DD 0003C .MOVQ 36(TCB), -(SP) : 1745
0000V CF C6 FB 0003E .PUSHL TCB : 1744
2C A6 D4 00043 .CALLS #6, FILL_MAP
79 11 00046 .CLRL 44(TCB) : 1750
6E D4 00048 2$: .BRB 11$ : 1754
22 AE 010E 8F B0 0004A .CLRL CUR_BUF_SIZ
24 AE 04 AE 9E 00050 .MOVW #270, B0FFER+2 : 1763
02 6C 91 00055 .MOVAB TEMP_BUF, BUFFER+4 : 1765
1E 12 00058 .CMPB (AP), #2 : 1771
5E DD 0005A .BNEQ 4$
08 AE DD 0005C .PUSHL SP : 1774
7E 08 AC 3C 0005F .PUSHL BUFFER+4 : 1775
7E 04 AC 3C 00063 .MOVZWL COL_NO, -(SP) : 1774
0000V CF 05 FB 00067 .MOVZWL LINE_NO, -(SP)
53 50 D0 0006E .PUSHL TCB
04 53 E8 00071 .CALLS #5, SET_CURSOR
50 53 D0 00074 3$: .MOVL R0, STATUS
04 00077 .BLBS STATUS, 4$ : 1776
00 52 CF 00078 4$: .MOVL STATUS, R0
0022 03 000E 0008 0007C 5$: .RET : 1778
57 D4 00084 6$: .CASEL TERM_TYPE, #0, #3
50 D4 00086 .WORD 6$-5$, -
1C 11 00088 7$: 7$-5$, -
05 D0 0008A 7$: 8$-5$, -
50 FF60 CF 9E 0008D .CLRL ERASE_LEN : 1783
MOVAB P.AAG, ERASE_PTR : 1784
BRB 10$ : 1778
MOVAB #5, ERASE_LEN : 1789
MOVAB P.AAG, ERASE_PTR : 1790
```

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

```
; Routine Size: 197 bytes,    Routine Base: _LIB$CODE + 03A3
```

: 1577 1820 1 !<BLF/PAGE>

```
1579 1821 1 %SBTTL 'SCR$PUT_BUFFER - Put Current Buffer to Screen or Prev. Buffer'
1580 1822 1 GLOBAL ROUTINE SCR$PUT_BUFFER (
1581 1823 1     OLD_BUFFER
1582 1824 1 ) =
1583 1825 1
1584 1826 1 ++
1585 1827 1 FUNCTIONAL DESCRIPTION:
1586 1828 1     This routine is the converse of SCR$SET_BUFFER. It puts the
1587 1829 1     contents of the current buffer to another buffer or the screen
1588 1830 1     if no buffer is given.
1589 1831 1
1590 1832 1 CALLING SEQUENCE:
1591 1833 1     ret_status.wlc.v = SCR$PUT_BUFFER (OLD_BUFFER.rv.r)
1592 1834 1
1593 1835 1 FORMAL PARAMETERS:
1594 1836 1     OLD_BUFFER.rv.r      Optional. Address of the previous
1595 1837 1                          buffer to put the current buffer into.
1596 1838 1                          If not specified or zero, put current
1597 1839 1                          buffer to the screen.
1598 1840 1
1599 1841 1 IMPLICIT INPUTS:
1600 1842 1     NONE
1601 1843 1
1602 1844 1 IMPLICIT OUTPUTS:
1603 1845 1     NONE
1604 1846 1
1605 1847 1 COMPLETION STATUS:
1606 1848 1     SSS_NORMAL          Normal successful completion
1607 1849 1
1608 1850 1 SIDE EFFECTS:
1609 1851 1     The current buffer is output and reinitialized to empty. If
1610 1852 1     OLD_BUFFER is nonzero, it is established to become the new
1611 1853 1     current buffer.
1612 1854 1
1613 1855 1 --
1614 1856 1
1615 1857 1 BEGIN
1616 1858 1
1617 1859 1 BUILTIN
1618 1860 1     NULLPARAMETER;
1619 1861 1
1620 1862 1 LOCAL
1621 1863 1     STATUS,                ! Status to be returned to caller
1622 1864 1
1623 1865 1     CURR_BUFF : REF BLOCK [, BYTE], ! Current contents of
1624 1866 1                                     ! TCB [SCR$L_BUFFER]
1625 1867 1
1626 1868 1     ADDR,                ! Address of newly-specified buffer or
1627 1869 1                             ! zero (indicating no further buffering)
1628 1870 1
1629 1871 1     TERM_TYPE,           ! dummy parameter - not used
1630 1872 1
1631 1873 1
1632 1874 1
1633 1875 1
1634 1876 1
1635 1877 1
```

```
1636 1878 2          TCB : REF BLOCK [, BYTE] ;          ! Pointer to current terminal
1637 1879 2          ! control block
1638 1880 2
1639 1881 2          STATUS = SCR$$GET_TYPE_R3 (-1; TCB, TERM_TYPE) ;
1640 1882 2          ! Set up TCB address
1641 1883 2          IF NOT .STATUS          ! -1 request - foreign terminal handler
1642 1884 2          ! doesn't do this one
1643 1885 2          THEN RETURN (.STATUS);
1644 1886 2
1645 1887 2          +
1646 1888 2          ! Determine future buffer address. It will be either the user-specified
1647 1889 2          ! buffer address or zero
1648 1890 2          -
1649 1891 2          ADDR = (IF NOT NULLPARAMETER (1)
1650 1892 2          THEN .OLD_BUFFER
1651 1893 2          ELSE 0) ;
1652 1894 2
1653 1895 2          +
1654 1896 2          ! Save the buffer address currently active.
1655 1897 2          -
1656 1898 2          CURR_BUFF = .TCB [SCR$L_BUFFER] ;
1657 1899 2
1658 1900 2          +
1659 1901 2          ! Save the future buffer address in the terminal control block
1660 1902 2          -
1661 1903 2          TCB [SCR$L_BUFFER] = .ADDR ;
1662 1904 2
1663 1905 2          +
1664 1906 2          ! If buffering was turned off, don't limit future output to buffered size.
1665 1907 2          -
1666 1908 2          IF .ADDR EQL 0
1667 1909 2          THEN
1668 1910 2          TCB [SCR$L_BUFSIZE] = BUFSIZE;
1669 1911 2
1670 1912 2          +
1671 1913 2          ! If there was no prior buffer, there is nothing to flush to the screen
1672 1914 2          -
1673 1915 2          IF .CURR_BUFF EQL 0
1674 1916 2          THEN
1675 1917 2          RETURN (SS$_NORMAL) ;
1676 1918 2
1677 1919 2          +
1678 1920 2          ! There previously was some buffering in effect, decide whether to
1679 1921 2          ! flush it to the screen or move it into the newly-specified buffer.
1680 1922 2          -
1681 1923 2          IF .TCB [SCR$L_CHARMAP] NEQ 0
1682 1924 2          THEN
1683 1925 2          BEGIN ! Mapping in effect
1684 1926 2          STATUS = SS$_NORMAL ; ! Assume success to follow
1685 1927 2          IF .ADDR EQL 0
1686 1928 2          THEN
1687 1929 2          BEGIN ! Buffering no longer in effect
1688 1930 2          STATUS = PUT_MAP (.TCB) ; ! Flush out what we've got
1689 1931 2          END; ! Buffering no longer in effect
1690 1932 2          END ! Mapping in effect
1691 1933 2          ELSE
1692 1934 2          BEGIN ! No mapping in effect
```

```

: 1693      1935  3      LOCAL
: 1694      1936  3      LOC_DESC : BLOCK [8, BYTE];
: 1695      1937  3      !+
: 1696      1938  3      !- Build a descriptor describing current contents of buffer
: 1697      1939  3      !-
: 1698      1940  3      LOC_DESC [DSC$W_LENGTH] = .CURR_BUFF [DSC$W_LENGTH] ;
: 1699      1941  3      LOC_DESC [DSC$B_CLASS] = DSC$K_CLASS_S ;
: 1700      1942  3      LOC_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;
: 1701      1943  3      LOC_DESC [DSC$A_POINTER] = .CURR_BUFF [DSC$A_POINTER] ;
: 1702      1944  3      !+
: 1703      1945  3      !- Reset buffer control block to make it look empty.
: 1704      1946  3      !-
: 1705      1947  3      CURR_BUFF [DSC$W_LENGTH] = 0 ;
: 1706      1948  3      !+
: 1707      1949  3      !- Output to previous buffer (or screen if TCB now contains 0)
: 1708      1950  3      !-
: 1709      1951  3      STATUS = OUTPUT ( .TCB, LOC_DESC ) ;
: 1710      1952  3      !-
: 1711      1953  3      END;      ! No mapping in effect
: 1712      1954  2
: 1713      1955  2
: 1714      1956  2      RETURN (.STATUS);
: 1715      1957  1      END;      ! End of routine SCR$PUT_BUFFER
```

			001C 00000	.ENTRY	SCR\$PUT_BUFFER, Save R2,R3,R4	1822
5E		08	C2 00002	SUBL2	#8, SP	
50		01	CE 00005	MNEGL	#1, R0	1881
		0000V	30 00008	BSBW	SCR\$GET_TYPE_R3	
54		51	D0 0000B	MOVL	R1, R4	
5B		50	E9 0000E	BLBC	STATUS, 6\$	1883
		6C	95 00011	TSTB	(AP)	1891
		0B	13 00013	BEQL	1\$	
	04	AC	D5 00015	TSTL	4(AP)	
		06	13 00018	BEQL	1\$	
51	04	BC	D0 0001A	MOVL	@OLD_BUFFER, ADDR	1892
		02	11 0001E	BRB	2\$	
		51	D4 00020 1\$:	CLRL	ADDR	1891
52	04	A4	D0 00022 2\$:	MOVL	4(TCB), CURR_BUFF	1898
04	A4	51	D0 00026	MOVL	ADDR, 4(TCB)	1903
		53	D4 0002A	CLRL	R3	1908
		51	D5 0002C	TSTL	ADDR	
		0B	12 0002E	BNEQ	3\$	
		53	D6 00030	INCL	R3	
4C	A4	0200	8F 3C 00032	MOVZWL	#512, 76(TCB)	1910
		52	D5 00038 3\$:	TSTL	CURR_BUFF	1915
		04	12 0003A	BNEQ	4\$	
50		01	D0 0003C	MOVL	#1, R0	1917
		04	0003F	RET		
	18	A4	D5 00040 4\$:	TSTL	24(TCB)	1923
		0E	13 00043	BEQL	5\$	
50		01	D0 00045	MOVL	#1, STATUS	1926
21		53	E9 00048	BLBC	R3, 6\$	1927
		54	DD 0004B	PUSHL	TCB	1930

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

SCR\$PUT_BUFFER - Put Current Buffer to Screen o

M 2
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 47
(17)

0000V CF 01 FB 0004D
04 00052
02 6E 62 B0 00053 5\$:
04 AE 010E 8F B0 00056
AE 04 A2 D0 0005C
62 B4 00061
4010 8F BB 00063
0000V CF 02 FB 00067
04 0006C 6\$:

CALLS #1, PUT_MAP
RET
MOVW (CURR_BUFF), LOC_DESC
MOVW #270, LOC_DESC+2
MOVL 4(CURR_BUFF), LOC_DESC+4
CLRW (CURR_BUFF)
PUSHR #*M<R4, SP>
CALLS #2, OUTPUT
RET

: 1923
: 1940
: 1942
: 1943
: 1948
: 1953
: 1957

; Routine Size: 109 bytes, Routine Base: _LIB\$CODE + 0468

; 1716 1958 1 !<BLF/PAGE>

```
1718 1959 1 %SBTTL 'SCR$PUT_LINE - Put Text to Screen in Line Mode'
1719 1960 1 GLOBAL ROUTINE SCR$PUT_LINE (
1720 1961 1     TEXT      : REF BLOCK [,BYTE],
1721 1962 1     LINE_ADV,
1722 1963 1     FLAGS
1723 1964 1 ) =
1724 1965 1 ++
1725 1966 1 FUNCTIONAL DESCRIPTION:
1726 1967 1
1727 1968 1     This routine is used to write messages to the terminal
1728 1969 1     followed by cursor movement sequences.
1729 1970 1
1730 1971 1 CALLING SEQUENCE:
1731 1972 1
1732 1973 1     ret_status.wlc.v = SCR$PUT_LINE (TEXT.rt.dx
1733 1974 1     [,LINE_ADV.rl.v,
1734 1975 1     [,FLAGS.rl.v])
1735 1976 1
1736 1977 1 FORMAL PARAMETERS:
1737 1978 1
1738 1979 1     TEXT.rt.dx      Address of descriptor of output string.
1739 1980 1
1740 1981 1     LINE_ADV.rl.v   Optional. Signed number of lines to advance
1741 1982 1     after output.
1742 1983 1
1743 1984 1     FLAGS.rl.v     Optional. Rendition codes.
1744 1985 1     If omitted, SCR$M_NORMAL will be used.
1745 1986 1     Values:
1746 1987 1         SCR$M_BLINK    display characters blinking.
1747 1988 1         SCR$M_BOLD     display characters in
1748 1989 1         higher-than-normal intensity.
1749 1990 1         SCR$M_NORMAL   display characters using
1750 1991 1         rendition associated with
1751 1992 1         window.
1752 1993 1         SCR$M_REVERSE  display characters in reverse
1753 1994 1         video -- i.e., using opposite
1754 1995 1         rendition from window default.
1755 1996 1         SCR$M_UNDERLINE display characters underlined.
1756 1997 1
1757 1998 1 IMPLICIT INPUTS:
1758 1999 1
1759 2000 1     NONE
1760 2001 1
1761 2002 1 IMPLICIT OUTPUTS:
1762 2003 1
1763 2004 1     NONE
1764 2005 1
1765 2006 1 COMPLETION STATUS:
1766 2007 1
1767 2008 1     SS$ _NORMAL      Normal successful completion
1768 2009 1
1769 2010 1 SIDE EFFECTS:
1770 2011 1
1771 2012 1     NONE
1772 2013 1 --
1773 2014 1
1774 2015 2 BEGIN
```

```
1775 2016 2
1776 2017 2
1777 2018 2 BUILTIN
1778 2019 2 ACTUALCOUNT;
1779 2020 2
1780 2021 2 MACRO
1781 2022 2 VT05_CR = %STRING (%CHAR (CR))%,
1782 2023 2 VT52_CR = %STRING (%CHAR (CR))%,
1783 2024 2 VT100_CR = %STRING (%CHAR (CR))%;
1784 2025 2
1785 2026 2 LITERAL
1786 2027 2 K_LINE_ARG = 2, ! Line_adv is arg 2 if present
1787 2028 2 K_MAX_ARGS = 3; ! 3 args if all present
1788 2029 2
1789 2030 2 LOCAL
1790 2031 2 TERM_TYPE, ! either type or devtyp from TCB
1791 2032 2 STATUS, ! status ret'd from called routine
1792 2033 2 TCB, ! ptr to terminal control block
1793 2034 2 TMP_LINE_ADV, ! temp loc for line advance
1794 2035 2 TMP_FLAGS; ! temp loc for rendition flags
1795 2036 2
1796 2037 2 MAP
1797 2038 2 TCB : REF BLOCK [,BYTE];
1798 2039 2
1799 2040 2 !+
1800 2041 2 ! Save the terminal type from the Terminal Control Block based on
1801 2042 2 ! whether this is a DEC terminal or foreign terminal.
1802 2043 2 !-
1803 2044 2 STATUS = SCR$$GET_TYPE_R3 (SCR$C_PUT_LINE; TCB, TERM_TYPE);
1804 2045 2 ! get current TCB
1805 2046 2 IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTForeign
1806 2047 2 THEN RETURN (.STATUS);
1807 2048 2
1808 2049 2 !+
1809 2050 2 ! Validate the number of arguments passed.
1810 2051 2 !-
1811 2052 2
1812 2053 2 IF ACTUALCOUNT () EQL 0 OR
1813 2054 2 ACTUALCOUNT () GTRU K_MAX_ARGS
1814 2055 2 THEN
1815 2056 2 RETURN (LIB$_WRONUMARG);
1816 2057 2
1817 2058 2 IF ACTUALCOUNT () LSS K_MAX_ARGS
1818 2059 2 THEN
1819 2060 2 TMP_FLAGS = SCR$M_NORMAL ! 3rd arg omitted - use default
1820 2061 2 ELSE
1821 2062 2 TMP_FLAGS = .FLAGS; ! use caller's rendition codes
1822 2063 2
1823 2064 2 IF ACTUALCOUNT () LSS K_LINE_ARG
1824 2065 2 THEN
1825 2066 2 TMP_LINE_ADV = 1 ! 2nd arg omitted - use default
1826 2067 2 ELSE
1827 2068 2 TMP_LINE_ADV = .LINE_ADV; ! use caller's lines to advance
1828 2069 2
1829 2070 2 !+
1830 2071 2 ! Call SCR$PUT_SCREEN to output the text and then call SCR$UP_SCROLL
1831 2072 2 ! or SCR$DOWN_SCROLL to position it.
```

; R

; 2

```
1832 2073 2  !-
1833 2074 2
1834 2075 2 STATUS = SCR$PUT_SCREEN (.TEXT, 0, 0, .TMP_FLAGS);
1835 2076 2 IF NOT .STATUS THEN RETURN (.STATUS);
1836 2077 2
1837 2078 2 IF .TMP_LINE_ADV LSS 0
1838 2079 2 THEN
1839 2080 2 BEGIN
1840 2081 3 INCR COUNTER FROM 1 TO (-.TMP_LINE_ADV) DO
1841 2082 4 BEGIN
1842 2083 4 STATUS = SCR$DOWN_SCROLL ();
1843 2084 4 IF NOT .STATUS THEN RETURN (.STATUS);
1844 2085 3 END;
1845 2086 3 END ! negative line adv - down scroll
1846 2087 2 ELSE
1847 2088 2 BEGIN
1848 2089 3 INCR COUNTER FROM 1 TO .TMP_LINE_ADV DO
1849 2090 4 BEGIN
1850 2091 4 STATUS = SCR$UP_SCROLL ();
1851 2092 4 IF NOT .STATUS THEN RETURN (.STATUS);
1852 2093 3 END;
1853 2094 2 END; ! positive line adv - scroll up
1854 2095 2
1855 2096 2 TCB [SCR$L_COLUMN] = 1; ! reset cursor to col 1
1856 2097 2
1857 2098 2 !+
1858 2099 2 ! Determine if a <CR> should be output.
1859 2100 2 !-
1860 2101 2
1861 2102 2 IF .TERM_TYPE NEQ 0 OR ! known terminal
1862 2103 2 .TCB [SCR$L_BUFFER] EQL 0 ! buffering not enabled
1863 2104 2 THEN
1864 2105 2 BEGIN ! <CR> needed
1865 2106 2 LOCAL
1866 2107 2 BUFFER : BLOCK [8, BYTE]; ! buffer dsc for OUTPUT
1867 2108 2
1868 2109 2 BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1869 2110 2 BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
1870 2111 2 BUFFER [DSC$W_LENGTH] = 1;
1871 2112 2
1872 2113 2 CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
1873 2114 2 SET
1874 2115 2
1875 2116 2 [UNKNOWN]:
1876 2117 2 BEGIN
1877 2118 2 BUFFER [DSC$W_LENGTH] = 0;
1878 2119 2 BUFFER [DSC$A_POINTER] = 0;
1879 2120 2 END;
1880 2121 2
1881 2122 2 [VT05]:
1882 2123 2 BUFFER [DSC$A_POINTER] = UPLIT (BYTE (VT05_CR));
1883 2124 2
1884 2125 2 [VT52]:
1885 2126 2 BUFFER [DSC$A_POINTER] = UPLIT (BYTE (VT52_CR));
1886 2127 2
1887 2128 2 [VT100]:
1888 2129 2 BUFFER [DSC$A_POINTER] = UPLIT (BYTE (VT100_CR));
```

```
: 1889      2130  3
: 1890      2131  3      [INRANGE,OUTRANGE]:
: 1891      2132  3      RETURN 0;          ! should never get here
: 1892      2133  3
: 1893      2134  3      TES;
: 1894      2135  3
: 1895      2136  3      STATUS = OUTPUT (.TCB, BUFFER); ! output a <CR>
: 1896      2137  3      IF NOT .STATUS THEN RETURN (.STATUS);
: 1897      2138  2      END;
: 1898      2139  2
: 1899      2140  2      RETURN (SS$_NORMAL);
: 1900      2141  1      END;          ! End of routine SCR$PUT_LINE
```

```
                                004D5      .BLKB      3
                                0D 004D8 P.AAJ: .ASCII <13>
                                004D9      .BLKB      3
                                0D 004DC P.AAK: .ASCII <13>
                                004DD      .BLKB      3
                                0D 004E0 P.AAL: .ASCII <13>

                                00FC 00000      .ENTRY SCR$PUT_LINE, Save R2,R3,R4,R5,R6,R7
57      FD73      CF 9E 00002      MOVAB SCR$DOWN_SCROLL, R7
5E      08 C2 00007      SUBL2 #8, SP
50      07 D0 0000A      MOVL #7, R0
                                0000V 30 0000D      BSBW SCR$$GET_TYPE_R3
54      51 D0 00010      MOVL R1, R4
5F      50 E9 00013      BLBC STATUS, 12$
04      OA A4 91 00016      CMPB 10(TCB), #4
                                01 12 0001A      BNEQ 1$
                                04 0001C      RET
                                6C 95 0001D 1$: TSTB (AP)
                                05 13 0001F      BEQL 2$
03      6C 91 00021      CMPB (AP), #3
                                08 1B 00024      BLEQU 3$
50 00000000G 8F D0 00026 2$: MOVL #LIB$_WRONUMARG, R0
                                04 0002D      RET
                                04 1E 0002E 3$: BGEQU 4$
                                51 D4 C0030      CLRL TMP_FLAGS
                                04 11 00032      BRB 5$
51      0C AC D0 00034 4$: MOVL FLAGS, TMP_FLAGS
02      6C 91 00038 5$: CMPB (AP), #2
                                05 1E 0003B      BGEQU 6$
53      01 D0 0003D      MOVL #1, TMP_LINE_ADV
                                04 11 00040      BRB 7$
53      08 AC D0 00042 6$: MOVL LINE_ADV, TMP_LINE_ADV
                                51 D0 00046 7$: PUSHL TMP_FLAGS
                                7E 7C 00048      CLRL -(SP)
                                04 AC D0 0004A      PUSHL TEXT
0000V CF 04 FB 0004D      CALLS #4, SCR$PUT_SCREEN
78      50 E9 00052      BLBC STATUS, 24$
                                53 D5 00055      TSTL TMP_LINE_ADV
                                13 1B 00057      BGEQ 10$
56      53 CE 00059      MNEGL TMP_LINE_ADV, R6
```

			55	D4	0005C	CLRL	COUNTER	:		:	2		
			06	11	0005E	BRB	9\$	:		:	2		
	67		00	FB	00060	8\$:	CALLS	#0, SCR\$DOWN_SCROLL	:	2083	:	2	
	67		50	E9	00063	BLBC	STATUS, 24\$	:	2084	:	2		
F6	55		56	F3	00066	9\$:	AOBLEQ	R6, COUNTER, 8\$	:	2081	:	2	
			10	11	0006A	BRB	14\$	:	2078	:	2		
			55	D4	0006C	10\$:	CLRL	COUNTER	:	2089	:	2	
			08	11	0006E	BRB	13\$	:		:	2		
	0000V	CF	00	FB	00070	11\$:	CALLS	#0, SCR\$UP_SCROLL	:	2091	:	2	
		55	50	E9	00075	2\$:	BLBC	STATUS, 24\$	:	2092	:	2	
F4		55	53	F3	00078	13\$:	AOBLEQ	TMP_LINE_ADV, COUNTER, 11\$	:	2089	:	2	
	28	A4	01	D0	0007C	14\$:	MOVL	#1, -40(TCB)	:	2096	:	2	
			52	D5	00080		TSTL	TERM_TYPE	:	2102	:	2	
			05	12	00082		BNEQ	15\$	:		:	2	
			04	A4	D5	00084	TSTL	4(TCB)	:	2103	:	2	
			3E	12	00087		BNEQ	22\$	:		:	2	
		6E	010E0001	8F	D0	00089	15\$:	MOVL	#17694721, BUFFER	:	2111	:	2
0021		00	52	CF	00090		CASEL	TERM_TYPE, #0, #3	:	2113	:	2	
	0019	0011	000A		00094	16\$:	.WORD	17\$-16\$,-	:		:	2	
								18\$-16\$,-	:		:	2	
								19\$-16\$,-	:		:	2	
								20\$-16\$	:		:	2	
			2D	11	0009C		BRB	23\$	:	2132	:		
			6E	B4	0009E	17\$:	CLRW	BUFFER	:	2118	:		
			04	AE	D4	000A0	CLRL	BUFFER+4	:	2119	:		
			16	11	000A3		BRB	21\$	:	2113	:		
	04	AE	FF4E	CF	9E	000A5	18\$:	MOVAB	P.AAJ, BUFFER+4	:	2123	:	
			0E	11	000AB		BRB	21\$	:		:		
	04	AE	FF4A	CF	9E	000AD	19\$:	MOVAB	P.AAK, BUFFER+4	:	2126	:	
			06	11	000B3		BRB	21\$	:		:		
	04	AE	FF46	CF	9E	000B5	20\$:	MOVAB	P.AAL, BUFFER+4	:	2129	:	
			4010	8F	BB	000BB	21\$:	PUSHR	#M<R4, SP>	:	2136	:	
	0000V	CF	02	FB	000BF		CALLS	#2, OUTPUT	:		:		
		06	50	E9	000C4		BLBC	STATUS, 24\$	:	2137	:		
		50	01	D0	000C7	22\$:	MOVL	#1, R0	:	2140	:		
				04	000CA		RET		:		:		
			50	D4	000CB	23\$:	CLRL	R0	:	2141	:		
			04	000CD	24\$:	RET			:		:		

; Routine Size: 206 bytes, Routine Base: _LIB\$CODE + 04E1

; 1901 2142 1 !<BLF/PAGE>

```
1903 2143 1 %SBTTL 'SCR$PUT_SCREEN - Put Text to Screen'
1904 2144 1 GLOBAL ROUTINE SCR$PUT_SCREEN (
1905 2145 1     TEXT      : REF BLOCK [,BYTE],
1906 2146 1     LINE_NO,
1907 2147 1     COL_NO,
1908 2148 1     FLAGS
1909 2149 1 ) =
1910 2150 1 ++
1911 2151 1 FUNCTIONAL DESCRIPTION:
1912 2152 1
1913 2153 1     This routine is used to write messages to the terminal.
1914 2154 1
1915 2155 1 CALLING SEQUENCE:
1916 2156 1
1917 2157 1     ret_status.wlc.v = SCR$PUT_SCREEN (TEXT.rt.dx
1918 2158 1                                     [,LINE_NO.rl.v,
1919 2159 1                                     COL_NO.rl.v]
1920 2160 1                                     [,FLAGS.rl.v])
1921 2161 1
1922 2162 1 FORMAL PARAMETERS:
1923 2163 1
1924 2164 1     TEXT.rt.dx      Address of descriptor of output string.
1925 2165 1
1926 2166 1     LINE_NO.rl.v    Optional. Line number at which to start output.
1927 2167 1                     If omitted (=0), the current line number is
1928 2168 1                     used.
1929 2169 1
1930 2170 1     COL_NO.rl.v     Optional. Column number at which to start
1931 2171 1                     output. If omitted (=0), the current line
1932 2172 1                     number is used.
1933 2173 1
1934 2174 1     FLAGS.rl.v      Optional. Rendition codes.
1935 2175 1                     If omitted, SCR$M_NORMAL will be used.
1936 2176 1                     Values:
1937 2177 1                         SCR$M_BLINK      display characters blinking.
1938 2178 1                         SCR$M_BOLD      display characters in
1939 2179 1                                         higher-than-normal intensity.
1940 2180 1                         SCR$M_NORMAL    display characters using
1941 2181 1                                         rendition associated with
1942 2182 1                                         window.
1943 2183 1                         SCR$M_REVERSE   display characters in reverse
1944 2184 1                                         video -- i.e., using opposite
1945 2185 1                                         rendition from window default.
1946 2186 1                         SCR$M_UNDERLINE display characters underlined.
1947 2187 1
1948 2188 1 IMPLICIT INPUTS:
1949 2189 1
1950 2190 1     NONE
1951 2191 1
1952 2192 1 IMPLICIT OUTPUTS:
1953 2193 1
1954 2194 1     NONE
1955 2195 1
1956 2196 1 COMPLETION STATUS:
1957 2197 1
1958 2198 1     SS$_NORMAL      Normal successful completion
1959 2199 1
```

```
1960 2200 1 | SIDE EFFECTS:
1961 2201 1 |
1962 2202 1 |     NONE
1963 2203 1 | --
1964 2204 1 |
1965 2205 2 |     BEGIN
1966 2206 2 |
1967 2207 2 |     BUILTIN
1968 2208 2 |         ACTUALCOUNT,
1969 2209 2 |         ACTUALPARAMETER,
1970 2210 2 |         NULLPARAMETER;
1971 2211 2 |
1972 2212 2 | +
1973 2213 2 | $SCR_APPEND
1974 2214 2 | Macro $SCR_APPEND appends the specified source string into the specified
1975 2215 2 | output buffer. It assumes that IN_POINTER points into a buffer
1976 2216 2 | BUFFER whose overall length is .TCB [SCR$L_BUFSIZ]
1977 2217 2 | It appends as much of the input string as will fit, and updates
1978 2218 2 | the buffer pointer by the number of bytes moved.
1979 2219 2 | -
1980 2220 2 | MACRO
1981 2221 2 | $SCR_APPEND ( IN_LENGTH, IN_POINTER, OUT_POINTER) =
1982 2222 2 | BEGIN
1983 2223 2 | LOCAL
1984 2224 2 |     AMT_TO_MOVE;      ! No. bytes that will fit
1985 2225 2 |
1986 2226 2 |     AMT_TO_MOVE = IN_LENGTH ;      ! Assume all will fit
1987 2227 2 |
1988 2228 2 | +
1989 2229 2 |     ! If space remaining is less than space needed, limit move to
1990 2230 2 |     ! amount that left.
1991 2231 2 | -
1992 2232 2 |     IF .TCB [SCR$L_BUFSIZ] - (.OUT_POINTER - .BUFFER) ! space remaining
1993 2233 2 |     LSS IN_LENGTH                                     ! space needed
1994 2234 2 | THEN
1995 2235 2 |     AMT_TO_MOVE = MAX (0, .TCB [SCR$L_BUFSIZ] - (.OUT_POINTER - .BUFFER));
1996 2236 2 |                                     ! don't allow a negative length
1997 2237 2 |
1998 2238 2 |     CH$MOVE ( .AMT_TO_MOVE,      ! length
1999 2239 2 |               IN_POINTER,        ! source addr
2000 2240 2 |               .OUT_POINTER);     ! dest. addr
2001 2241 2 |
2002 2242 2 |     OUT_POINTER = .OUT_POINTER + .AMT_TO_MOVE ; ! Advance pointer
2003 2243 2 | END;
2004 2244 2 | % ;      ! End of macro $SCR_APPEND
2005 2245 2 |
2006 2246 2 | LOCAL
2007 2247 2 |     TCB : REF BLOCK [, BYTE],      ! Pointer to current terminal
2008 2248 2 |                                     ! control block
2009 2249 2 |     TYPE,                          ! device type
2010 2250 2 |     STATUS,                        ! Status to return to caller
2011 2251 2 |     LOC_DESC : BLOCK [8, BYTE];    ! Local descriptor
2012 2252 2 |
2013 2253 2 | +
2014 2254 2 |     ! Set up pointer to current terminal control block
2015 2255 2 | -
2016 2256 2 |     STATUS = SCR$GET_TYPE_R3 (SCR$C_PUT_SCREEN; TCB, TYPE) ;
```



```
2017 2257 2 IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTFORIGN
2018 2258 2 THEN RETURN (.STATUS);
2019 2259 2
2020 2260 2 !+
2021 2261 2 ! See if mapping or buffering is in effect
2022 2262 2 !-
2023 2263 2 IF .TCB [SCR$B_CHARMAP] EQL 0 OR ! If no mapping
2024 2264 2 .TCB [SCR$B_BUFFER] EQL 0 ! or no buffering
2025 2265 2 THEN
2026 2266 3 BEGIN ! No buffering or no mapping or neither
2027 2267 3
2028 2268 3 IF ACTUALCOUNT() EQL 1
2029 2269 3 THEN
2030 2270 4 BEGIN ! Only 1 arg
2031 2271 4 LOCAL
2032 2272 4 LOC_DESC : BLOCK [8, BYTE]; ! Local descriptor
2033 2273 4 !+
2034 2274 4 ! Extract length and address of callers text. If descriptor
2035 2275 4 ! has unrecognized class code, quit. Otherwise, build
2036 2276 4 ! local fixed-length descriptor to describe text string
2037 2277 4 ! to OUTPUT for outputting.
2038 2278 4 !-
2039 2279 5 IF NOT ( STATUS = LIB$ANALYZE_SDESC_R2 (!Get len. and addr
2040 2280 5 .TEXT;
2041 2281 5 LOC_DESC [DSC$B_LENGTH],
2042 2282 5 LOC_DESC [DSC$B_POINTER]))
2043 2283 4 THEN
2044 2284 4 RETURN (.STATUS);
2045 2285 4
2046 2286 4 !+
2047 2287 4 ! Complete descriptor and do output
2048 2288 4 !-
2049 2289 4 LOC_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
2050 2290 4 LOC_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2051 2291 4
2052 2292 4 STATUS = OUTPUT (.TCB, LOC_DESC);
2053 2293 4 END ! Only 1 arg
2054 2294 3 ELSE
2055 2295 4 BEGIN ! More than one arg
2056 2296 4 LOCAL
2057 2297 4 LENGTH, ! Length of callers text string
2058 2298 4 ADDR, ! Address of callers text string
2059 2299 4 BUFFER : REF BLOCK [, BYTE], ! Pointer to allocated
2060 2300 4 ! buffer
2061 2301 4 BUF_PTR, ! Pointer into BUFFER
2062 2302 4 ! as it is filled
2063 2303 4 ATTR_GIVEN; ! Flag to record
2064 2304 4 ! whether an attribute
2065 2305 4 ! setting was placed in
2066 2306 4 ! the buffer and a "turn
2067 2307 4 ! off graphic" rendition
2068 2308 4 ! needs to be inserted
2069 2309 4 ! at the end of buffer.
2070 2310 5 IF NOT (STATUS = LIB$GET_VM ( TCB [SCR$B_BUFSIZ], ! length
2071 2311 5 BUFFER)) ! returned addr
2072 2312 4 THEN
2073 2313 4 RETURN (.STATUS);
```

```
: 2074      2314  4
: 2075      2315  4      BUF_PTR = .BUFFER ; ! Initialize to front of buffer
: 2076      2316  4
: 2077      2317  4      ATTR_GIVEN = 0;      ! Assume we won't be outputting
: 2078      2318  4      ! attribute sequence.
: 2079      2319  4      IF NOT NULLPARAMETER (4)      ! FLAGS present
: 2080      2320  4      THEN
: 2081      2321  5          BEGIN      ! Flags present
: 2082      2322  5
: 2083      2323  5          CASE .TYPE FROM UNKNOWN TO VTFOREIGN OF
: 2084      2324  5          SET
: 2085      2325  5              [UNKNOWN,
: 2086      2326  5              VT05,
: 2087      2327  5              VT52,
: 2088      2328  5              VTFOREIGN]:
: 2089      2329  5              0 ;      ! Do nothing
: 2090      2330  5
: 2091      2331  5          [INRANGE, OTRANGE]:
: 2092      2332  5              RETURN ( LIB$_FATERRLIB ) ;
: 2093      2333  5
: 2094      2334  5          [VT100]:
: 2095      2335  6              BEGIN      ! VT100 processing
: 2096      2336  6              +
: 2097      2337  6              ! Make sure MBZ bits in high end of longword
: 2098      2338  6              ! are in fact zero.
: 2099      2339  6              -
: 2100      2340  6              IF .FLAGS<16,16> NEQ 0
: 2101      2341  6              THEN
: 2102      2342  6                  RETURN ( LIB$_INVARG); ! Undefined bits set
: 2103      2343  6
: 2104      2344  6              IF .FLAGS<0,4> NEQ 0
: 2105      2345  6              THEN
: 2106      2346  7                  BEGIN      ! Attributes set
: 2107      2347  7                  ATTR_GIVEN = 1 ;      ! Record that we will be
: 2108      2348  7                  ! outputting attribute
: 2109      2349  7                  ! escape sequence
: 2110      2350  7                  CH$WCHAR_A ( ESC, BUF_PTR ) ;
: 2111      2351  7                  CH$WCHAR_A ( LB, BUF_PTR ) ;
: 2112      2352  7                  +
: 2113      2353  7                  ! For each attribute bit set in the flags
: 2114      2354  7                  ! parameter, copy the appropriate ASCII
: 2115      2355  7                  ! graphic rendition byte followed by a ";"
: 2116      2356  7                  ! into our temporary buffer BUFFER.
: 2117      2357  7                  ! Note use of autoincrementing.
: 2118      2358  7                  -
: 2119      2359  7                  INCR I FROM 0 TO 3
: 2120      2360  7                  DO
: 2121      2361  8                      BEGIN      ! Build attribute string
: 2122      2362  8                      BIND ATTRTABL = UPLIT ( BYTE ( '1754' ) )
: 2123      2363  8                      : VECTOR [4, BYTE];
: 2124      2364  8                      IF .FLAGS<.I, 1>
: 2125      2365  8                      THEN
: 2126      2366  9                          BEGIN
: 2127      2367  9                          CH$WCHAR_A ( .ATTRTABL [.I],
: 2128      2368  9                          BUF_PTR);
: 2129      2369  9                          CH$WCHAR_A ( '%C';',', BUF_PTR);
: 2130      2370  8                      END;
```

: R
: 2

```
2131 2371 7
2132 2372 7
2133 2373 7
2134 2374 7
2135 2375 7
2136 2376 7
2137 2377 7
2138 2378 7
2139 2379 7
2140 2380 7
2141 2381 6
2142 2382 5
2143 2383 5
2144 2384 4
2145 2385 4
2146 2386 4
2147 2387 4
2148 2388 4
2149 2389 4
2150 2390 4
2151 2391 4
2152 2392 5
2153 2393 5
2154 2394 5
2155 2395 5
2156 2396 5
2157 2397 4
2158 2398 4
2159 2399 5
2160 2400 5
2161 2401 5
2162 2402 4
2163 2403 4
2164 2404 4
2165 2405 4
2166 2406 4
2167 2407 4
2168 2408 4
2169 2409 4
2170 2410 4
2171 2411 4
2172 2412 5
2173 2413 5
2174 2414 5
2175 2415 5
2176 2416 5
2177 2417 5
2178 2418 5
2179 2419 4
2180 2420 4
2181 2421 4
2182 2422 4
2183 2423 4
2184 2424 4
2185 2425 4
2186 2426 4
2187 2427 4

      END;      . Build attribute string
      +
      When we fall out of loop above we have
      deposited an extra ";" at the end of the
      buffer. Back up BUF_PTR so that deposit
      of VT100_SGR lands on top of it.
      -
      BUF_PTR = .BUF_PTR - 1;
      CH$QCHAR_A ( VT100_SGR, BUF_PTR );

      END;      ! Attributes set
      END;      ! VT100 processing
      TES;
      END;      ! Flags present

      +
      If cursor position was specified on call,
      invoke SET_CURSOR to get us there.
      -
      IF NOT NULLPARAMETER (2)
      THEN
      BEGIN
      LOCAL
      RET_LEN : INITIAL (0);
      SET_CURSOR ( .TCB, .LINE_NO, .COL_NO, .BUF_PTR, RET_LEN );
      BUF_PTR = .BUF_PTR + .RET_LEN;
      END;

      IF NOT ( STATUS = LIB$ANALYZE_SDESC_R2 ( .TEXT;
      LENGTH,
      ADDR))
      THEN
      RETURN ( .STATUS );

      +
      Insert callers text
      -
      $SCR_APPEND ( .LENGTH, .ADDR, BUF_PTR );

      IF .ATTR_GIVEN
      THEN
      BEGIN      ! Insert VT100_OFF
      MACRO
      VT100_OFF = %STRING ( %CHAR(ESC), %CHAR(LB), '0m')%;
      $SCR_APPEND ( %CHARCOUNT ( VT100_OFF ),
      UPLIT ( BYTE ( VT100_OFF ) ),
      BUF_PTR );

      END;      ! Insert VT100_OFF

      +
      Build local descriptor referencing the intermediate buffer
      we've been constructing and output it.
      -
      LOC_DESC [DSC$W_LENGTH] = .BUF_PTR - .BUFFER;
      LOC_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
      LOC_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
```

```
2188 2428 4      LOC_DESC [DSC$A_POINTER] = .BUFFER;
2189 2429 4      STATUS = OUTPUT( .TCB, LOC_DESC );
2190 2430 4
2191 2431 4      !+
2192 2432 4      ! Give back buffer space we've been using
2193 2433 4      !-
2194 2434 4      LIB$FREE_VM ( TCB [SCR$S_BUFSIZ], BUFFER );
2195 2435 4
2196 2436 3      END;          ! More than one arg
2197 2437 3
2198 2438 3      END          ! No buffering or no mapping or neither
2199 2439 3
2200 2440 2      ELSE
2201 2441 2
2202 2442 3      BEGIN      ! Buffering and mapping active
2203 2443 3      LOCAL
2204 2444 3          LENGTH,      ! Length of callers string
2205 2445 3          ADDR,        ! Address of callers string
2206 2446 3          INDEX,      ! Linear index into character buffer
2207 2447 3          CURR_LINE,   ! Line at which to set cursor
2208 2448 3          CURR_COL;    ! Column at which to set cursor
2209 2449 3      !+
2210 2450 3      ! Set up cursor arguments. Assume the ones from the TCB but
2211 2451 3      ! overwrite if user has specified.
2212 2452 3      !-
2213 2453 3      CURR_LINE = .TCB [SCR$S_LINE] ;
2214 2454 3      CURR_COL = .TCB [SCR$S_COLUMN] ;
2215 2455 3      IF NOT NULLPARAMETER (2) AND      ! If line_no supplied
2216 2456 3      NOT NULLPARAMETER (3)          ! and col_no supplied
2217 2457 3      THEN
2218 2458 4          BEGIN      ! User-supplied cursor position
2219 2459 4          CURR_LINE = .LINE_NO ;
2220 2460 4          CURR_COL = .COL_NO ;
2221 2461 4          !+
2222 2462 4          ! Reset cursor position in TCB to new values.
2223 2463 4          !-
2224 2464 4          TCB [SCR$S_LINE] = .LINE_NO ;
2225 2465 4          TCB [SCR$S_COLUMN] = .COL_NO ;
2226 2466 3          END;      ! User-supplied cursor position
2227 2467 3
2228 2468 3      !+
2229 2469 3      ! If specified starting line is beyond the bottom of the
2230 2470 3      ! screen, quit immediately, but call it a success.
2231 2471 3      !-
2232 2472 3
2233 2473 3      IF .CURR_LINE GTR .TCB [SCR$S_DEVPAGSIZ]
2234 2474 3      THEN
2235 2475 3          RETURN ( SSS_NORMAL ) ;
2236 2476 3
2237 2477 3      !+
2238 2478 3      ! Calculate zero-based linear index into buffer
2239 2479 3      !-
2240 2480 3      INDEX = (.CURR_LINE - 1) * .TCB [SCR$S_DEVWIDTH] +
2241 2481 3      (.CURR_COL - 1) ;
2242 2482 3
2243 2483 3      !+
2244 2484 3      ! Extract length and address of callers text string. If
```

```
2245 2485 3      | unrecognized class of descriptor, quit.
2246 2486 3      |
2247 2487 4      | IF NOT (STATUS = LIB$ANALYZE_SDESC_R2 ( .TEXT; LENGTH, ADDR))
2248 2488 3      | THEN
2249 2489 3      |     RETURN ( .STATUS ) ;
2250 2490 3      |
2251 2491 3      | +
2252 2492 3      | | If length of users text reaches beyond end of buffer, quit
2253 2493 3      | |
2254 2494 3      | | IF .LENGTH + .INDEX GTR .TCB [SCR$L_AREA]
2255 2495 3      | | THEN
2256 2496 3      | |     RETURN ( LIB$_SCRBUFOVF ) ;
2257 2497 3      | |
2258 2498 3      | |
2259 2499 3      | | TCB [SCR$L_COLUMN] = .TCB [SCR$L_COLUMN] + .LENGTH ; !***???***
2260 2500 3      | |
2261 2501 3      | | +
2262 2502 3      | | | Move users text into mapping buffer
2263 2503 3      | | |
2264 2504 3      | | | CH$MOVE ( .LENGTH, .ADDR, .TCB [SCR$L_CHARMAP] + .INDEX ) ;
2265 2505 3      | | |
2266 2506 3      | | | +
2267 2507 3      | | | | Check for attributes for this text and set up our attribute
2268 2508 3      | | | | mask.
2269 2509 3      | | | |
2270 2510 3      | | | |
2271 2511 3      | | | | TCB [SCR$L_ATTRMASK] = .TCB [SCR$L_ATTRMASK] OR
2272 2512 4      | | | |     (IF ACTUALCOUNT ( ) EQL 4 ! flag arg passed
2273 2513 4      | | | |     THEN
2274 2514 4      | | | |     (NOT (.TCB [SCR$L_DEVCHAR])) AND .FLAGS
2275 2515 4      | | | |     ELSE
2276 2516 3      | | | |     SCR$_NORMAL);
2277 2517 3      | | | |
2278 2518 3      | | | | +
2279 2519 3      | | | | | If any attributes turned on, fill them into the attribute map
2280 2520 3      | | | | |
2281 2521 3      | | | | | IF .TCB [SCR$L_ATTRMASK] NEQ 0
2282 2522 3      | | | | | THEN
2283 2523 4      | | | | |     CH$FILL ( (IF ACTUALCOUNT ( ) EQL 4
2284 2524 4      | | | | |     THEN .FLAGS
2285 2525 3      | | | | |     ELSE SCR$_NORMAL),
2286 2526 3      | | | | |     .LENGTH,
2287 2527 3      | | | | |     .TCB [SCR$L_ATTRMAP] + .INDEX) ;
2288 2528 3      | | | | |
2289 2529 3      | | | | | STATUS = SS$_NORMAL ;
2290 2530 2      | | | | | END;      ! Buffering and mapping active
2291 2531 2      | | | | |
2292 2532 2      | | | | | RETURN (SS$_NORMAL);
2293 2533 1      | | | | | END;      ! End of routine SCR$PUT_SCREEN
```

```
34 35 37 31 005AF .BLKB 1
6D 30 5B 1B 005B0 P.AAM: .ASCII \1754\
005B4 P.AAN: .ASCII <27>\[Om\

ATTRTABL= P.AAM
```

			OFFC	00000	.ENTRY	SCR\$PUT_SCREEN, Save R2,R3,R4,R5,R6,R7,R8,-	
						R9,R10,R11	2144
					SUBL2	#24, SP	
					CLRL	R0	2256
					BSBW	SCR\$\$GET_TYPE_R3	
					MOVL	R0, STATUS	
					MOVL	R1, R7	
					MOVL	R2, R4	
					BLBC	STATUS, 1\$	2257
					CMPB	10(TCB), #4	
					BNEQ	2\$	
					BRW	20\$	
					TSTL	24(TCB)	2263
					BEQL	3\$	
					TSTL	4(TCB)	2264
					BEQL	3\$	
					BRW	18\$	
					CMPB	(AP), #1	2268
					BNEQ	4\$	
					MOVL	TEXT, R0	2281
					JSB	LIB\$ANALYZE_SDESC_R2	
					MOVL	R0, STATUS	
					MOVW	R1, LOC_DESC	
					MOVL	R2, LOC_DESC+4	2282
					BLBC	STATUS, 1\$	2281
					MOVW	#270, LOC_DESC+2	2290
					PUSHAB	LOC_DESC	2292
					PUSHL	TCB	
					CALLS	#2, OUTPUT	
					MOVL	R0, STATUS	
					BRW	28\$	2268
					PUSHAB	BUFFER	2310
					PUSHAB	76(TCB)	
					CALLS	#2, LIB\$GET_VM	
					MOVL	R0, STATUS	
					BLBC	STATUS, 1\$	
					MOVL	BUFFER, R10	2315
					MOVL	R10, BUF_PTR	
					CLRL	ATTR_GIVEN	2317
					CMPB	(AP), #4	2319
					BLSSU	10\$	
					TSTL	16(AP)	
					BEQL	10\$	
					CASEL	TYPE, #0, #4	2323
					.WORD	10\$-5\$,-	
						10\$-5\$,-	
						10\$-5\$,-	
						6\$-5\$,-	
						10\$-5\$	
					MOVL	#LIB\$_FATERRLIB, R0	2332
					RET		
					TSTW	FLAGS+2	2340
					BEQL	7\$	
					MOVL	#LIB\$_INVARG, R0	2342
					RET		

		OF	10	AC	93	000A8	7\$:	BITB	FLAGS, #15	2344
				22	13	000AC		BEQL	10\$	
		5B		01	D0	000AE		MOVL	#1, ATTR_GIVEN	2347
		86	5B1B	8F	B0	000B1		MOVW	#23323, (BUF_PTR)+	2350
				50	D4	000B6		CLRL	I	2359
09	10	AC		50	E1	000B8	8\$:	BBC	I, FLAGS, 9\$	2364
		86	FF36	CF	40	000BD		MOVB	ATTRTABL[I], (BUF_PTR)+	2368
		86		3B	90	000C3		MOVB	#59, (BUF_PTR)+	2369
EE		50		03	F3	000C6	9\$:	AOBLEQ	#3, I, 8\$	2359
		76	6D	8F	90	000CA		MOVB	#109, -(BUF_PTR)	2379
				56	D6	000CE		INCL	BUF_PTR	
		02		6C	91	000D0	10\$:	CMPB	(APT, #2	2390
				19	1F	000D3		BLSSU	11\$	
			08	AC	D5	000D5		TSTL	8(AP)	
				14	13	000D8		BEQL	11\$	
				6E	D4	000DA		CLRL	RET_LEN	2392
			4040	8F	BB	000DC		PUSHR	#*MZR6, SP>	2395
		7E	08	AC	7D	000E0		MOVQ	LINE_NO, -(SP)	
				57	DD	000E4		PUSHL	TCB	
	0000V	CF		05	FB	000E6		CALLS	#5, SET_CURSOR	
		56		6E	C0	000EB		ADDL2	RET_LEN, BUF_PTR	2396
		50	04	AC	D0	000EE	11\$:	MOVL	TEXT, R0	2399
			00000000G	00	16	000F2		JSB	LIB\$ANALYZE_SDESC_R2	
		59		50	D0	000F8		MOVL	R0, STATUS	
		03		59	E8	000FB		BLBS	STATUS, 12\$	
				00BF	31	000FE		BRW	20\$	
		58		51	D0	00101	12\$:	MOVL	LENGTH, AMT TO MOVE	2408
50		5A		56	C3	00104		SUBL3	BUF_PTR, R10, R0	
		50	4C	A7	C0	00108		ADDL2	76(TCB), R0	
		51		50	D1	0010C		CMPL	R0, LENGTH	
				09	18	0010F		BGEQ	14\$	
				50	D5	00111		TSTL	R0	
				02	18	00113		BGEQ	13\$	
				50	D4	00115		CLRL	R0	
		58		50	D0	00117	13\$:	MOVL	R0, AMT TO MOVE	
66		62		58	28	0011A	14\$:	MOVC3	AMT TO MOVE, (ADDR), (BUF_PTR)	
		56		58	C0	0011E		ADDL2	AMT TO MOVE, BUF_PTR	
		26		5B	E9	00121		BLBC	ATTR_GIVEN, 17\$	2410
		58		04	D0	00124		MOVL	#4, AMT TO MOVE	2417
50		56		5A	C3	00127		SUBL3	R10, BUF_PTR, R0	
		51	04	A0	9E	0012B		MOVAB	4(R0), RT	
		51	4C	A7	D1	0012F		CMPL	76(TCB), R1	
				0C	18	00133		BGEQ	16\$	
51	4C	A7		50	C3	00135		SUBL3	R0, 76(TCB), R1	
				02	18	0013A		BGEQ	15\$	
				51	D4	0013C		CLRL	R1	
		58		51	D0	0013E	15\$:	MOVL	R1, AMT TO MOVE	
66	FEB6	CF		58	28	00141	16\$:	MOVC3	AMT TO MOVE, P.AAN, (BUF_PTR)	
		56		58	C0	00147		ADDL2	AMT TO MOVE, BUF_PTR	
10	AE	56		5A	A3	0014A	17\$:	SUBW3	R10, BUF_PTR, LOC_DESC	2425
		12	AE	8F	B0	0014F		MOVW	#270, LOC_DESC+2	2427
		14	AE	5A	D0	00155		MOVL	R10, LOC_DESC+4	2428
				10	AE	9F	00159	PUSHAB	LOC_DESC	2429
				57	DD	0015C		PUSHL	TCB	
	0000V	CF		02	FR	0015E		CALLS	#2, OUTPUT	
		59		50	D0	00163		MOVL	R0, STATUS	
			04	AE	9F	00166		PUSHAB	BUFFER	2434

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SCR\$PUT_SCREEN - Put Text to Screen

B 4
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 62
(19)

LIB
V04

00000000G	00	4C	A7	9F	00169	PUSHAB	76(TCB)	:	
			02	FB	0016C	CALLS	#2, LIB\$FREE_VM	:	
	50	24	0097	31	00173	BRW	28\$	:	2263
	02		A7	7D	00176	18\$: MOVQ	36(TCB), CURR_LINE	:	2453
			6C	91	0017A	CMPB	(AP), #2	:	2455
		08	18	1F	0017D	BLSSU	19\$	:	
			AC	D5	0017F	TSTL	8(AP)	:	
	03		13	13	00182	BEQL	19\$	:	
			6C	91	00184	CMPB	(AP), #3	:	2456
			0E	1F	00187	BLSSU	19\$	:	
		0C	AC	D5	00189	TSTL	12(AP)	:	
			09	13	0018C	BEQL	19\$	:	
	50	08	AC	7D	0018E	MOVQ	LINE_NO, CURR_LINE	:	2459
	A7	08	AC	7D	00192	MOVQ	LINE_NO, 36(TCB)	:	2464
50	0E	A7	24	10	00	ED	00197	19\$:	2473
				6E	19	0019D	CMPZV	#0, #16, 14(TCB), CURR_LINE	
				50	D7	0019F	BLSS	28\$	2480
	52	0C	A7	3C	001A1	DECL	R0	:	
	50		52	C4	001A5	MOVZWL	12(TCB), R2	:	
	56	FF	A140	9E	001A8	MULL2	R2, R0	:	
	50	04	AC	D0	001AD	MOVAB	-1(CURR COL)[R0], INDEX	:	
		00000000G	00	16	001B1	MOVL	TEXT, R0	:	2487
	59		50	D0	001B7	JSB	LIB\$ANALYZE_SDESC_R2	:	
	58		51	D0	001BA	MOVL	R0, STATUS	:	
	04		59	E8	001BD	MOVL	R1, R8	:	
	50		59	D0	001C0	BLBS	STATUS, 21\$	:	2489
				04	001C3	MOVL	STATUS, R0	:	
	51		58	C1	001C4	RET		:	2494
	14		A7	D1	001C8	ADDL3	INDEX, LENGTH, R1	:	
				08	15	001CC	CMPL	R1, 20(TCB)	
	50	00000000G	8F	D0	001CE	BLEQ	22\$	:	
				04	001D5	MOVL	#LIB\$_SCRBUFOVF, R0	:	2496
	28		A7	C0	001D6	RET		:	
			62	58	28	001DA	ADDL2	LENGTH, 40(TCB)	2499
18 B746	04		6C	91	001E0	MOVCL3	LENGTH, (ADDR), @24(TCB)[INDEX]	:	2504
			08	12	001E3	CMPB	(AP), #4	:	2512
50	10	AC	10	A7	CB	001E5	BNEQ	23\$	
				02	11	001EB	BICL3	16(TCB), FLAGS, R0	2514
				50	D4	001ED	BRB	24\$	
	2C	A7	50	C8	001EF	23\$: CLRL	R0	:	2512
			15	13	001F3	24\$: BISL2	R0, 44(TCB)	:	
	04		6C	91	001F5	BEQL	27\$	:	2521
			06	12	001F8	CMPB	(AP), #4	:	2523
	50	10	AC	D0	001FA	BNEQ	25\$	:	
			02	11	001FE	MOVL	FLAGS, R0	:	2524
			50	D4	00200	BRB	26\$	:	
58	50	6E	1C	00	2C	00202	25\$: CLRL	R0	2523
				01	D0	0020A	26\$: MOVCL5	#0, (SP), R0, LENGTH, @28(TCB)[INDEX]	2527
				01	D0	0020D	27\$: MOVL	#1, STATUS	2529
				04	00210	28\$: MOVL	#1, R0	:	2532
							RET	:	2533

; Routine Size: 529 bytes, Routine Base: _LIB\$CODE + 05B8

; 2294 2534 1 !<BLF/PAGE>


```
: 2296 2535 1 %SBTTL 'SCR$SET_BUFFER - Set/Clear Buffer Mode'
: 2297 2536 1 GLOBAL ROUTINE SCR$SET_BUFFER (
: 2298 2537 1     BUFFER      : REF BLOCK [, BYTE],
: 2299 2538 1     OLD_BUFFER : REF VECTOR [, LONG],
: 2300 2539 1 ) =
: 2301 2540 1 ++
: 2302 2541 1   FUNCTIONAL DESCRIPTION:
: 2303 2542 1
: 2304 2543 1       This routine is called to establish a buffer to be used to hold
: 2305 2544 1       terminal output rather than immediately writing the output to
: 2306 2545 1       SYS$OUTPUT. It is also called to turn off the buffering mode.
: 2307 2546 1       The buffer address is established such that all screen output
: 2308 2547 1       is appended to the end of the buffer.
: 2309 2548 1
: 2310 2549 1   CALLING SEQUENCE:
: 2311 2550 1
: 2312 2551 1       ret_status.wlc.v = SCR$SET_BUFFER (BUFFER.mt.dx
: 2313 2552 1                                     [,OLD_BUFFER.wl.r])
: 2314 2553 1
: 2315 2554 1   FORMAL PARAMETERS:
: 2316 2555 1
: 2317 2556 1       BUFFER.mt.dx    Address of descriptor of buffer.  If 0,
: 2318 2557 1                       buffering mode is turned off.
: 2319 2558 1
: 2320 2559 1       OLD_BUFFER.wl.r Optional. Address of the location which will
: 2321 2560 1                       contain the address of the previous buffer.
: 2322 2561 1
: 2323 2562 1   IMPLICIT INPUTS:
: 2324 2563 1
: 2325 2564 1       NONE
: 2326 2565 1
: 2327 2566 1   IMPLICIT OUTPUTS:
: 2328 2567 1
: 2329 2568 1       NONE
: 2330 2569 1
: 2331 2570 1   COMPLETION STATUS:
: 2332 2571 1
: 2333 2572 1       SS$_NORMAL      Normal successful completion
: 2334 2573 1
: 2335 2574 1   SIDE EFFECTS:
: 2336 2575 1
: 2337 2576 1       NONE
: 2338 2577 1 --
: 2339 2578 1
: 2340 2579 2   BEGIN
: 2341 2580 2
: 2342 2581 2   BUILTIN
: 2343 2582 2       ACTUALCOUNT,
: 2344 2583 2       ACTUALPARAMETER,
: 2345 2584 2       NULLPARAMETER;
: 2346 2585 2
: 2347 2586 2   LOCAL
: 2348 2587 2       TCB : REF BLOCK [, BYTE],      : Pointer to current terminal
: 2349 2588 2                                     : control block.
: 2350 2589 2       TERM TYPE,                     : dummy parameter - device type
: 2351 2590 2       STATUS;
: 2352 2591 2
```

```
: 2353 2592 2 LITERAL
: 2354 2593 2 K_MAX_ARGS = 2; ! # of args if all present
: 2355 2594 2
: 2356 2595 2 !+
: 2357 2596 2 ! Get current device type and set up our pointer to the terminal
: 2358 2597 2 ! control block.
: 2359 2598 2 !-
: 2360 2599 2 STATUS = SCR$$GET_TYPE_R3 (-1; TCB, TERM_TYPE);
: 2361 2600 2 ! -1 request - foreign terminal handler
: 2362 2601 2 ! doesn't do the action
: 2363 2602 2 IF NOT .STATUS THEN RETURN (.STATUS);
: 2364 2603 2
: 2365 2604 2 !+
: 2366 2605 2 ! If no arguments provided, error.
: 2367 2606 2 !-
: 2368 2607 2 IF ACTUALCOUNT () LEQ 0 OR
: 2369 2608 2 ACTUALCOUNT () GTR K_MAX_ARGS
: 2370 2609 2 THEN
: 2371 2610 2 RETURN ( LIB$_INVARG ); ! *** Should be LIB$_WRONUMARG
: 2372 2611 2
: 2373 2612 2 !+
: 2374 2613 2 ! If 2nd parameter supplied, return current buffer address from TCB
: 2375 2614 2 ! before changing it.
: 2376 2615 2 !-
: 2377 2616 2 IF NOT NULLPARAMETER (2)
: 2378 2617 2 THEN
: 2379 2618 2 OLD_BUFFER [0] = .TCB [SCR$L_BUFFER];
: 2380 2619 2
: 2381 2620 2 !+
: 2382 2621 2 ! See if a new buffer address is being provided, or whether caller is
: 2383 2622 2 ! just cancelling buffering.
: 2384 2623 2 !-
: 2385 2624 2 IF NOT NULLPARAMETER (1)
: 2386 2625 2 THEN
: 2387 2626 3 BEGIN ! New buffer descriptor address provided
: 2388 2627 3 LOCAL
: 2389 2628 3 STATUS, ! Error status to return to caller if need be
: 2390 2629 3 LENGTH : WORD, ! Length of buffer provided by caller
: 2391 2630 3 ADDR : REF VECTOR [,LONG];
: 2392 2631 3 ! Address of buffer provided by caller. We
: 2393 2632 3 ! declare it as a vector of longwords since
: 2394 2633 3 ! we need to write control information into the
: 2395 2634 3 ! 1st 3 longwords. The rest of the screen
: 2396 2635 3 ! routines think of it as a block of bytes.
: 2397 2636 3
: 2398 2637 4 IF NOT (STATUS = LIB$ANALYZE_SDESC_R2 ( .BUFFER; LENGTH, ADDR ))
: 2399 2638 3 THEN
: 2400 2639 3 RETURN (.STATUS); ! Bad descriptor class
: 2401 2640 3
: 2402 2641 3 !+
: 2403 2642 3 ! Length of buffer provided must be greater than 12 bytes or
: 2404 2643 3 ! we can't really use it since we use the 1st 12 bytes to
: 2405 2644 3 ! store control information.
: 2406 2645 3 !-
: 2407 2646 3 IF .LENGTH LEQ 12
: 2408 2647 3 THEN
: 2409 2648 3 RETURN ( LIB$_SCRBUFOVF );
```

```
2410 2649 3
2411 2650 3
2412 2651 3
2413 2652 3
2414 2653 3
2415 2654 3
2416 2655 3
2417 2656 3
2418 2657 3
2419 2658 3
2420 2659 3
2421 2660 3
2422 2661 2
2423 2662 3
2424 2663 3
2425 2664 3
2426 2665 2
2427 2666 2
2428 2667 2
2429 2668 1

      *
      * Initialize control information at top of buffer provided.
      *
      ADDR [0] = 0 ;      ! Used length set to zero
      ADDR [1] = ADDR [3] ; ! Address of start of usable buffer area
                           ! beyond control information
      ADDR [2] = .LENGTH - 12 ; ! Adjust length of remaining
                           ! buffer
      TCB [SCR$L_BUFFER] = ADDR [0] ; ! Address of new buffer
      TCB [SCR$L_BUFSIZE] = .ADDR [2] ; ! Save size of buffer used
      END      ! New buffer descriptor address provided
ELSE
  BEGIN      ! Cancel buffering
    TCB [SCR$L_BUFFER] = 0 ;
    TCB [SCR$L_BUFSIZE] = BUFSIZE;
  END;      ! Cancel buffering

RETURN (SS$_NORMAL);
END;      ! End of routine SCR$SET_BUFFER
```

50	01	CE	00002	.ENTRY	SCR\$SET_BUFFER, Save R2,R3,R4	2536
	0000V	30	00005	MN\$GL	#1, R0	2599
54	51	D0	00008	BSBW	SCR\$GET_TYPE_R3	
66	50	E9	00008	MOVL	R1, R4	
	6C	95	0000E	BLBC	STATUS, 7\$	2602
	05	13	00010	TSTB	(AP)	2607
02	6C	91	00012	BEQL	1\$	
	08	1B	00015	CMPB	(AP), #2	2608
50 00000000G	8F	D0	00017	BLEQU	2\$	
	04	0001E		MOVL	#LIB\$_INVARG, R0	2610
	0A	1F	0001F	RET		
	08	AC	D5 00021	BLSSU	3\$	2616
	05	13	00024	TSTL	8(AP)	
08 BC	04	A4	D0 00026	BEQL	3\$	
	6C	95	0002B	MOVL	4(TCB), @OLD_BUFFER	2618
	39	13	0002D	TSTB	(AP)	2624
	04	AC	D5 0002F	BEQL	5\$	
	34	13	00032	TSTL	4(AP)	
50	04	AC	D0 00034	BEQL	5\$	
33 00000000G	00	16	00038	MOVL	BUFFER, R0	2637
0C	50	E9	0003E	JSB	LIB\$ANALYZE_SDESC_R2	
	51	B1	00C41	BLBC	STATUS, 7\$	
	08	1A	00044	CMPW	LENGTH, #12	2646
50 00000000G	8F	D0	00046	BGTRU	4\$	
	04	0004D		MOVL	#LIB\$_SCRBUFOVF, R0	2648
	62	D4	0004E	RET		
04 A2	0C	A2	9E 00050	CLRL	(ADDR)	2653
08 A2		51	3C 00055	MOVAB	12(R2), 4(ADDR)	2654
08 A2		0C	C2 00059	MOVZWL	LENGTH, 8(ADDR)	2656
04 A4		52	D0 0005D	SUBL2	#12, 8(ADDR)	
4C A4	08	A2	D0 00061	MOVL	ADDR, 4(TCB)	2658
				MOVL	8(ADDR), 76(TCB)	2659

LIB\$SCREEN - Screen Management
SCR\$SET_BUFFER - Set/Clear Buffer Mode

F 4
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

```
VAX-11 Bliss-32 V4.0-742
[VM$LIB.SRC]SCREEN.B32;1
```

Page 66
(20)

LIB
V04

			09	11	00066		BRB	6\$
		04	A4	D4	00068	5\$:	CLRL	4(TCB)
4C	A4	0200	8F	3C	00068		MOVZWL	#512, 76(TCB)
	50		01	D0	00071	6\$:	MOVL	#1, R0
				04	00074	7\$:	RET	

; 2624
 ; 2663
 ; 2664
 ; 2667
 ; 2668

```
; Routine Size: 117 bytes,    Routine Base: _LIB$CODE + 07C9
```

; 2430 2669 1 !<BLF/PAGE>

.....

```
2432 2670 1 %SBTTL 'SCR$SET_CURSOR - Set Cursor to Character Position on Screen'
2433 2671 1 GLOBAL ROUTINE SCR$SET_CURSOR (
2434 2672 1     LINE_NO : WORD,
2435 2673 1     COL_NO  : WORD
2436 2674 1 ) =
2437 2675 1 ++
2438 2676 1 FUNCTIONAL DESCRIPTION:
2439 2677 1
2440 2678 1     This routine causes the cursor to be moved to the specified
2441 2679 1     position.
2442 2680 1
2443 2681 1 CALLING SEQUENCE:
2444 2682 1
2445 2683 1     ret_status.wlc.v = SCR$SET_CURSOR (LINE_NO.rw.v, COL_NO.rw.v)
2446 2684 1
2447 2685 1 FORMAL PARAMETERS:
2448 2686 1
2449 2687 1     LINE_NO.rw.v    Line number.
2450 2688 1
2451 2689 1     COL_NO.rw.v     Column number.
2452 2690 1
2453 2691 1 IMPLICIT INPUTS:
2454 2692 1
2455 2693 1     NONE
2456 2694 1
2457 2695 1 IMPLICIT OUTPUTS:
2458 2696 1
2459 2697 1     NONE
2460 2698 1
2461 2699 1 COMPLETION STATUS:
2462 2700 1
2463 2701 1     $$$_NORMAL      Normal successful completion
2464 2702 1
2465 2703 1 SIDE EFFECTS:
2466 2704 1
2467 2705 1     NONE
2468 2706 1 --
2469 2707 1
2470 2708 2 BEGIN
2471 2709 2
2472 2710 2 LITERAL
2473 2711 2     K_CURSOR_SIZ = 20;
2474 2712 2
2475 2713 2
2476 2714 2
2477 2715 2
2478 2716 2 LOCAL
2479 2717 2     STATUS,
2480 2718 2     BUFFER : BLOCK [8, BYTE],
2481 2719 2     TCB : REF BLOCK [, BYTE],
2482 2720 2     TERM_TYPE,
2483 2721 2     CURSOR_BUF : VECTOR [K_CURSOR_SIZ, BYTE];
2484 2722 2
2485 2723 2 STATUS = SCR$$GET_TYPE_R3 (SCR$C_SET_CURSOR; TCB, TERM_TYPE);
2486 2724 2
2487 2725 2 IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL V$FOREIGN
2488 2726 2 THEN RETURN (.STATUS);
```

: R

: 2

```
2489 2727 2
2490 2728 2
2491 2729 2 !+
2492 2730 2 !- Just store new cursor position if mapping active.
2493 2731 2
2494 2732 2 TCB [SCR$L_LINE] = .LINE NO;
2495 2733 2 TCB [SCR$L_COLUMN] = .COL_NO;
2496 2734 2
2497 2735 2 IF .TCB [SCR$L_CHARMAP] NEQ 0
2498 2736 2 THEN
2499 2737 2     RETURN (SS$_NORMAL);           ! ret if mapping active
2500 2738 2
2501 2739 2 !+
2502 2740 2 !- Put the appropriate set cursor sequence into the buffer
2503 2741 2
2504 2742 2
2505 2743 2 BUFFER = 0 ; ! Preset 1st long word to zero, its used as a current
2506 2744 2             ! length in the call below. Fill in class and dtype
2507 2745 2             ! later
2508 2746 2 BUFFER [DSC$A_POINTER] = CURSOR_BUF [0];
2509 2747 2
2510 2748 2 IF NOT (STATUS = SET_CURSOR (.TCB, .LINE NO, .COL NO,
2511 2749 2     .BUFFER [DSC$A_POINTER],
2512 2750 2     BUFFER [DSC$W_LENGTH]))
2513 2751 2
2514 2752 2 THEN RETURN (.STATUS);
2515 2753 2
2516 2754 2 !+
2517 2755 2 !- Output the sequence.
2518 2756 2
2519 2757 2
2520 2758 2 BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2521 2759 2 BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
2522 2760 2
2523 2761 2 IF NOT (STATUS = OUTPUT (.TCB, BUFFER))
2524 2762 2 THEN RETURN (.STATUS);
2525 2763 2
2526 2764 2 RETURN (SS$_NORMAL);
2527 2765 1 END;           ! End of routine SCR$SET_CURSOR
```

```
001C 00000
5E 1C C2 00002
50 04 D0 00005
0000V 30 00008
54 51 D0 0000B
4A 50 E9 0000E
04 0A A4 91 00011
44 13 00015
24 A4 04 AC 3C 00017
28 A4 08 AC 3C 0001C
18 A4 D5 00021
32 12 00024
14 AE D4 00026
```

```
.ENTRY SCR$SET_CURSOR, Save R2,R3,R4
SUBL2 #28, SP
MOVL #4, R0
BSBW SCR$SET_TYPE_R3
MOVL R1, R4
BLBC STATUS, 2$
CMPB 10(TCB), #4
BEQL 2$
MOVZWL LINE NO, 34(TCB)
MOVZWL COL NO, 40(TCB)
TSTL 24(TCB)
BNEQ 1$
CLRL BUFFER
```

```
2671
2723
2725
2732
2733
2735
2743
```

LIB\$SCREEN - Screen Management
SCR\$SET_CURSOR - Set Cursor to Character Positi

16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 69
(21)

LIB
V04

18	AE		6E	9E	00029	MOVAB	CURSOR_BUF, BUFFER+4
		14	AE	9F	0002D	PUSHAB	BUFFER
		1C	AE	DD	00030	PUSHL	BUFFER+4
	7E	08	AC	3C	00033	MOVZWL	COL_NO, -(SP)
	7E	04	AC	3C	00037	MOVZWL	LINE_NO, -(SP)
			54	DD	0003B	PUSHL	TCB
0000V	CF		05	FB	0003D	CALLS	#5, SET_CURSOR
	16		50	E9	00042	BLBC	STATUS, 2\$
16	AE	010E	8F	B0	00045	MOVW	#270, BUFFER+2
		14	AE	9F	0004B	PUSHAB	BUFFER
			54	DD	0004E	PUSHL	TCB
0000V	CF		02	FB	00050	CALLS	#2, OUTPUT
	03		50	E9	00055	BLBC	STATUS, 2\$
	50		01	D0	00058	MOVL	#1, R0
			04	0005B	1\$:	RET	
					2\$:		

[illegible]

```
; Routine Size: 92 bytes,    Routine Base: _LIB$CODE + 083E
```

; 2528 2766 1 !<BLF/PAGE>

.....

```
: 2530 2767 1 %SBTTL 'SCR$SET_SCROLL - Establish Scrolling Region'
: 2531 2768 1 GLOBAL ROUTINE SCR$SET_SCROLL (
: 2532 2769 1     START_LINE : WORD,
: 2533 2770 1     END_LINE   : WORD
: 2534 2771 1     ) =
: 2535 2772 1
: 2536 2773 1 ++
: 2537 2774 1 FUNCTIONAL DESCRIPTION:
: 2538 2775 1     This routine establishes a scrolling by setting the internal
: 2539 2776 1     scrolling region parameters.  Issues the escape sequence that
: 2540 2777 1     establishes the region and preserves the cursor position.
: 2541 2778 1
: 2542 2779 1     If this routine is called with no arguments, scrolling is turned
: 2543 2780 1     off by setting the scrolling region parameters to zeroes.
: 2544 2781 1
: 2545 2782 1 CALLING SEQUENCE:
: 2546 2783 1     ret_status.wlc.v = SCR$SET_SCROLL ([START_LINE.rw.v,
: 2547 2784 1     END_LINE.rw.v])
: 2548 2785 1
: 2549 2786 1 FORMAL PARAMETERS:
: 2550 2787 1     [START_LINE.rw.v]     Optional - First line of scrolling region.
: 2551 2788 1     [END_LINE.rw.v]       Optional - Last line of scrolling region.
: 2552 2789 1
: 2553 2790 1 IMPLICIT INPUTS:
: 2554 2791 1     NONE
: 2555 2792 1
: 2556 2793 1 IMPLICIT OUTPUTS:
: 2557 2794 1     NONE
: 2558 2795 1
: 2559 2796 1 COMPLETION STATUS:
: 2560 2797 1     SSS_NORMAL           Normal successful completion
: 2561 2798 1
: 2562 2799 1 SIDE EFFECTS:
: 2563 2800 1     NONE
: 2564 2801 1
: 2565 2802 1 --
: 2566 2803 1
: 2567 2804 1 BEGIN
: 2568 2805 1
: 2569 2806 1 LITERAL
: 2570 2807 1     K_ALL_ARGS = 2;
: 2571 2808 1     K_SCROLL_BUFSIZ = 25;
: 2572 2809 1
: 2573 2810 2
: 2574 2811 2
: 2575 2812 2
: 2576 2813 2     # args if all present
: 2577 2814 2     size of buffer to hold
: 2578 2815 2     set scroll sequence
: 2579 2816 2     **** more than large enough for
: 2580 2817 2     **** currently known sequences -
: 2581 2818 2     **** may need to be increased
: 2582 2819 2     **** later
: 2583 2820 2
: 2584 2821 2 LOCAL
: 2585 2822 2     SET_MARGINS : VECTOR [1, 8] INITIAL (
: 2586 2823 2     _DSC$K_DTYPE_T ^24 + _DSC$K_CLASS_S ^16 + 14,
```

; R


```
2587 2824 2      UPLIT ( BYTE (ESC, '7', ESC, LB, '!UL;!UL',
2588 2825 2          VT100_SM, ESC, '8' ))
2589 2826 2
2590 2827 2      Dsc for VT100 seq to <save cursor>
2591 2828 2      <establish scrolling region>
2592 2829 2      <restore cursor>
2593 2830 2      STATUS,
2594 2831 2      BUFFER : BLOCK [8, BYTE],
2595 2832 2      TEMP_BUF : VECTOR [K_SCROLL_BUFSIZ, BYTE], ! buffer for set scroll seq
2596 2833 2      CVT_ARGS : VECTOR [2],
2597 2834 2      FAO_LEN : WORD,
2598 2835 2      TERM_TYPE,
2599 2836 2      TCB;
2600 2837 2
2601 2838 2      MAP
2602 2839 2          TCB : REF BLOCK [,BYTE];
2603 2840 2
2604 2841 2      BUILTIN
2605 2842 2          ACTUALCOUNT;
2606 2843 2
2607 2844 2      STATUS = SCR$GET_TYPE_R3 (SCR$C_SET_SCROLL; TCB, TERM_TYPE);
2608 2845 2      IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTFOREIGN
2609 2846 2      THEN RETURN (.STATUS);
2610 2847 2
2611 2848 2      !+
2612 2849 2      ! Determine if scrolling region should be established.
2613 2850 2      !-
2614 2851 2
2615 2852 2      IF ACTUALCOUNT () GEQU K_ALL_ARGS
2616 2853 2      THEN
2617 2854 2          TCB [SCR$V_SCROLL] = 1          ! set scrolling active if args passed
2618 2855 2      ELSE
2619 2856 2          TCB [SCR$V_SCROLL] = 0;          ! scrolling inactive if no args passed
2620 2857 2
2621 2858 2      IF .TCB [SCR$L_CHARMAP] NEQ 0 OR      ! mapping active
2622 2859 2          .TCB [SCR$B_TYPE] NEQ VT100 OR  ! allow only VT100 terminals
2623 2860 2          .SCR$AL_DEVDEPND2 [TT2$V_DECCRT] EQL 0
2624 2861 2          ! must a a dec_crt not just VT100 style
2625 2862 2      THEN
2626 2863 2          RETURN 1;
2627 2864 2
2628 2865 2      !+
2629 2866 2      ! Point a descriptor to the buffer which will hold the set scroll
2630 2867 2      ! sequence.
2631 2868 2      !-
2632 2869 2
2633 2870 2      BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2634 2871 2      BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
2635 2872 2      BUFFER [DSC$W_LENGTH] = K_SCROLL_BUFSIZ;
2636 2873 2      BUFFER [DSC$A_POINTER] = TEMP_BUF;
2637 2874 2
2638 2875 2      !+
2639 2876 2      ! Call $FAO to convert START_LINE and END_LINE to ASCII. Notice
2640 2877 2      ! that the buffer we just allocated is empty, so the current length
2641 2878 2      ! is 0 and the first free byte is the start address of the buffer.
2642 2879 2      !-
2643 2880 2
```

```
2644 2881 2 IF ACTUALCOUNT ( ) GEQU K_ALL_ARGS
2645 2882 2 THEN
2646 2883 2 BEGIN
2647 2884 2 CVT_ARGS [0] = .START_LINE;
2648 2885 2 CVT_ARGS [1] = .END_LINE;
2649 2886 2 END
2650 2887 2 ELSE
2651 2888 2 BEGIN
2652 2889 2 CVT_ARGS [0] = 0;
2653 2890 2 CVT_ARGS [1] = 0;
2654 2891 2 END;
2655 2892 2
2656 P 2893 2 STATUS = $FAOL (CTRSTR = SET_MARGINS, OUTLEN = FAO_LEN,
2657 2894 2 OUTBUF = .BUFFER, PRMLST = CVT_ARGS);
2658 2895 2 IF NOT .STATUS THEN RETURN (.STATUS);
2659 2896 2 BUFFER [DSC$W_LENGTH] = .FAO_LEN;
2660 2897 2
2661 2898 2 !+
2662 2899 2 ! Output the scrolling sequence.
2663 2900 2 !-
2664 2901 2
2665 2902 2 IF .TCB [SCR$L_RTNADDR] EQL 0
2666 2903 2 THEN
2667 2904 2 BEGIN
2668 2905 2 STATUS = $QIOW (EFN = .TCB [SCR$L_EFN],
2669 2906 2 CHAN = .TCB [SCR$L_CHAN],
2670 2907 2 FUNC = (IOS$WRITEVBLK OR IOS$M_CANCTRL),
2671 2908 2 P1 = .BUFFER [DSC$A_POINTER],
2672 2909 2 P2 = 0);
2673 2910 2 IF NOT .STATUS THEN RETURN (.STATUS);
2674 2911 2 END;
2675 2912 2
2676 2913 2 STATUS = OUTPUT (.TCB, BUFFER); ! output buffer
2677 2914 2
2678 2915 2 IF NOT .STATUS
2679 2916 2 THEN
2680 2917 2 RETURN (.STATUS)
2681 2918 2 ELSE
2682 2919 2 RETURN (SS$NORMAL);
2683 2920 2
2684 2921 1 END;
! End of routine SCR$SET_SCROLL
```

```
4C 55 21 3B 4C 5B 1B 0089A
55 21 0089C P.AAO: .BLKB 2
1B 72 0089D .BYTE 27
38 0089E .ASCII \7\
55 21 008A0 .BYTE 27, 91
1B 72 008A7 .ASCII \!UL;!UL\
38 008A9 .BYTE 114, 27
.ASCII \8\

.EXTRN SYSS$FAOL, SYSS$QIOW

.ENTRY SCR$SET_SCROLL, Save R2,R3,R4
SUBL2 #56, SP
MOVL #234946574, SET_MARGINS
```

```
: 2768
:
: 2810
```

34	AE	E2	AF	9E	0000D	MOVAB	P.AAO, SET_MARGINS+4	
	50		09	D0	00012	MOVL	#9, R0	2843
			0000V	30	00015	BSBW	SCR\$GET_TYPE_R3	
54			51	D0	00018	MOVL	R1, R4	
5D			50	E9	0001B	BLBC	STATUS, 6\$	2845
04		0A	A4	91	0001E	CMPB	10(TCB), #4	
			01	12	00022	BNEQ	1\$	
				04	00024	RET		
02			6C	91	00025	CMPB	(AP), #2	2852
			06	1F	00028	BLSSU	2\$	
40	A4		01	88	0002A	BISB2	#1, 64(TCB)	2854
			04	11	0002E	BRB	3\$	
40	A4		01	8A	00030	BICB2	#1, 64(TCB)	2856
		18	A4	D5	00034	TSTL	24(TCP	2858
			7D	12	00037	BNEQ	8\$	
		03	0A	A4	91	CMPB	10(TCB), #3	2859
			77	12	0003D	BNEQ	8\$	
6F 00000000G	00		05	51	0003F	BBC	#5, SCR\$AL_DEVDEPND2+3, 8\$	2860
28	AE	010E0019	8F	D0	00047	MOVL	#17694745, -BUFFER	2872
2C	AE	0C	AE	9E	0004F	MOVAB	TEMP_BUF, BUFFER+4	2873
	02		6C	91	00054	CMPB	(AP), #2	2881
			0C	1F	00057	BLSSU	4\$	
04	AE	04	AC	3C	00059	MOVZWL	START_LINE, CVT_ARGS	2884
08	AE	08	AC	3C	0005E	MOVZWL	END_LINE, CVT_ARGS+4	2885
			03	11	00063	BRB	5\$	2881
		04	AE	7C	00065	CLRQ	CVT_ARGS	2889
		04	AE	5	00068	PUSHAB	CVT_ARGS	2894
		2C	AE	9F	0006B	PUSHAB	BUFFER	
		08	AE	9F	0006E	PUSHAB	FAO_LEN	
		3C	AE	9F	00071	PUSHAB	SET_MARGINS	
00000000G	00		04	FB	00074	CALLS	#4, SYS\$FAOL	
	3B		50	E9	0007B	BLBC	STATUS, 9\$	2895
28	AE		6E	B0	0007E	MOVW	FAO_LEN, BUFFER	2896
		38	A4	D5	00082	TSTL	56(TCB)	2902
			22	12	00085	BNEQ	7\$	
			7E	7C	00087	CLRQ	-(SP)	2909
			7E	7C	00089	CLRQ	-(SP)	
			7E	D4	0008B	CLRL	-(SP)	
		40	AE	DD	0008D	PUSHL	BUFFER+4	
			7E	7C	00090	CLRQ	-(SP)	
			7E	D4	00092	CLRL	-(SP)	
	7E	70	8F	9A	00094	MOVZBL	#112, -(SP)	
	7E	08	A4	3C	00098	MOVZWL	8(TCB), -(SP)	
		48	A4	DD	0009C	PUSHL	72(TCB)	
00000000G	00		0C	FB	0009F	CALLS	#12, SYS\$QIOW	
	10		50	E9	000A6	BLBC	STATUS, 9\$	2910
		28	AE	9F	000A9	PUSHAB	BUFFER	2913
			54	DD	000AC	PUSHL	TCB	
0000V	CF		02	FB	000AE	CALLS	#2, OUTPUT	
	03		50	E9	000B3	BLBC	STATUS, 9\$	2915
	50		01	D0	000B6	MOVL	#1, R0	2919
			04	000B9	9\$:	RET		2921

; Routine Size: 186 bytes, Routine Base: _LIB\$CODE + 08AA

; 2685 2922 1 !<BLF/PAGE>

```
2687 2923 1 %SBTTL 'SCR$UP_SCROLL - Up Scroll, Move Cursor Down One Line'
2688 2924 1 GLOBAL ROUTINE SCR$UP_SCROLL =
2689 2925 1 ++
2690 2926 1 FUNCTIONAL DESCRIPTION:
2691 2927 1
2692 2928 1 This routine causes the cursor to be moved down 1 line to the
2693 2929 1 same column of the following line. If the cursor was on the
2694 2930 1 bottom line to begin with, it stays where it was, but all the
2695 2931 1 information on the screen appears to move up one line. The
2696 2932 1 information that was on the top line is lost and a blank line
2697 2933 1 appears at the bottom.
2698 2934 1
2699 2935 1 If a scrolling region is active ( on a VT100 or in mapping)
2700 2936 1 then the logic above applies to the top and bottom lines of
2701 2937 1 the scrolling region rather than the top and bottom line of
2702 2938 1 the screen. If an UP_SCROLL is performed on the bottom line of
2703 2939 1 the screen, or a DOWN_SCROLL is performed on the top line of the
2704 2940 1 screen, while a scrolling region is active then no screen
2705 2941 1 movement takes place unless the top or bottom line of the screen
2706 2942 1 corresponds to the top or bottom line of the scrolling region.
2707 2943 1
2708 2944 1 CALLING SEQUENCE:
2709 2945 1
2710 2946 1 ret_status.wlc.v = SCR$UP_SCROLL ( )
2711 2947 1
2712 2948 1 FORMAL PARAMETERS:
2713 2949 1
2714 2950 1 NONE
2715 2951 1
2716 2952 1 IMPLICIT INPUTS:
2717 2953 1
2718 2954 1 NONE
2719 2955 1
2720 2956 1 IMPLICIT OUTPUTS:
2721 2957 1
2722 2958 1 NONE
2723 2959 1
2724 2960 1 COMPLETION STATUS:
2725 2961 1
2726 2962 1 $$$_NORMAL Normal successful completion
2727 2963 1
2728 2964 1 SIDE EFFECTS:
2729 2965 1
2730 2966 1 NONE
2731 2967 1 --
2732 2968 1
2733 2969 1 BEGIN
2734 2970 2
2735 2971 2 MACRO
2736 2972 2 VT05_UPSCROLL = %STRING (%CHAR(LF), %CHAR(NULL), %CHAR(NULL), %CHAR(NULL))%, ! VT05 (w/1111)
2737 2973 2 VT52_UPSCROLL = %STRING (%CHAR(LF))%, ! VT52
2738 2974 2 VT100_UPSCROLL = %STRING (%CHAR(LF))%, ! VT100
2739 2975 2
2740 2976 2 LOCAL
2741 2977 2 STATUS, ! used for calls
2742 2978 2 TCB, ! terminal control block
2743 2979 2 UP_BUFF : BLOCK [8,BYTE], ! buffer dsc for OUTPUT
```

```
2744 2980 2      TERM_TYPE;                      ! terminal type
2745 2981 2
2746 2982 2      MAP
2747 2983 2          TCB : REF BLOCK [,BYTE];
2748 2984 2
2749 2985 2      STATUS = SCR$$GET_TYPE_R3 (SCR$C_UP_SCROLL; TCB, TERM_TYPE);
2750 2986 2          ! get current control block,
2751 2987 2      IF NOT .STATUS OR .TCB [SCR$B_TYPE] EQL VTFOREIGN
2752 2988 2      THEN RETURN (.STATUS);
2753 2989 2
2754 2990 2      !+
2755 2991 2      !- Update cursor position with down movement. Limit cursor to page.
2756 2992 2
2757 2993 2
2758 2994 2      TCB [SCR$L_LINE] = MIN (.TCB [SCR$L_LINE] + 1,
2759 2995 2          .TCB [SCR$W_DEVPAGSIZ]);
2760 2996 2
2761 2997 2      !+
2762 2998 2      !- Do nothing if mapping active and buffering enabled.
2763 2999 2
2764 3000 2
2765 3001 2      IF .TCB [SCR$L_CHARMAP] NEQ 0 AND
2766 3002 2      .TCB [SCR$L_BUFFER] NEQ 0
2767 3003 2      THEN
2768 3004 2          RETURN (SS$_NORMAL);
2769 3005 2
2770 3006 2      !+
2771 3007 2      !- Output scrolling sequence according to terminal type.
2772 3008 2
2773 3009 2
2774 3010 2      UP_BUFF [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2775 3011 2      UP_BUFF [DSC$B_CLASS] = DSC$K_CLASS_S;
2776 3012 2
2777 3013 2      CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
2778 3014 2          SET
2779 3015 2              [UNKNOWN] :
2780 3016 2                  BEGIN
2781 3017 2                      JP_BUFF [DSC$W_LENGTH] = 0;
2782 3018 2                      UP_BUFF [DSC$A_POINTER] = 0;
2783 3019 2                  END;
2784 3020 2
2785 3021 2              [VT05] :
2786 3022 2                  BEGIN
2787 3023 2                      UP_BUFF [DSC$W_LENGTH] = %CHARCOUNT (VT05_UPSCROLL);
2788 3024 2                      UP_BUFF [DSC$A_POINTER] = UPLIT (BYTE (VT05_UPSCROLL));
2789 3025 2                  END;
2790 3026 2
2791 3027 2              [VT52] :
2792 3028 2                  BEGIN
2793 3029 2                      UP_BUFF [DSC$W_LENGTH] = %CHARCOUNT (VT52_UPSCROLL);
2794 3030 2                      UP_BUFF [DSC$A_POINTER] = UPLIT (BYTE (VT52_UPSCROLL));
2795 3031 2                  END;
2796 3032 2
2797 3033 2              [VT100] :
2798 3034 2                  BEGIN
2799 3035 2                      UP_BUFF [DSC$W_LENGTH] = %CHARCOUNT (VT100_UPSCROLL);
2800 3036 2                      UP_BUFF [DSC$A_POINTER] = UPLIT (BYTE (VT100_UPSCROLL));
```

```
: 2801      3037 2      END:
: 2802      3038 2
: 2803      3039 2      TES:
: 2804      3040 2
: 2805      3041 2      RETURN (STATUS = OUTPUT (.TCB, UP_BUFF));
: 2806      3042 2
: 2807      3043 1      END;                                ! End of routine SCR$UP_SCROLL
```

```
00 00 00 0A 00964 P.AAP: .ASCII <10><0><0><0>
0A 00968 P.AAQ: .ASCII <10>
00969 .BLKB 3
0A 0096C P.AAR: .ASCII <10>
```

				000C	00000	.ENTRY	SCR\$UP_SCROLL, Save R2,R3	2924
	5E		08	C2	00002	SUBL2	#8, SP	
	50		0A	D0	00005	MOVL	#10, R0	2985
			0000V	30	00008	BSBW	SCR\$GET TYPE_R3	
	67		5C	E9	0000B	BLBC	STATUS, 9\$	2987
	04		0A	A1	91 0000E	CMPB	10(TCB), #4	
				61	13 00012	BEQL	9\$	
				01	C1 00014	ADDL3	#1, 36(TCB), R3	2994
53		OE	53	24	A1	00	ED 00019	2995
			A1	10		04	18 0001F	
						0E	A1 3C 00021	
			53	24	A1	53	D0 00025 1\$:	2994
						18	A1 D5 00029	3001
						09	13 0002C	
						04	A1 D5 0002E	3002
						04	13 00031	
						50	01 D0 00033	3004
							04 00036	
			02	AE	010E	8F	B0 00037 2\$:	3010
			00			52	CF 0003D	3013
0023		03				000F	0008 00041 3\$:	
							.WORD	
							4\$-3\$,-	
							5\$-3\$,-	
							6\$-3\$,-	
							7\$-3\$	
							UP_BUFF	3017
							UP_BUFF+4	3018
							BRB	3013
							8\$	
							#4, UP_BUFF	3023
							P.AAP, -UP_BUFF+4	3024
							BRB	3013
							8\$	
							#1, UP_BUFF	3029
							P.AAQ, -UP_BUFF+4	3030
							BRB	3013
							8\$	
							#1, UP_BUFF	3035
							P.AAR, -UP_BUFF+4	3036
							PUSHR	3041
							#*M<R1, SP5	
							CALLS	
							#2, OUTPUT	
							RET	3043

; Routine Size: 118 bytes, Routine Base: _LIB\$CODE + 096D

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SCR\$UP_SCROLL - Up Scroll, Move Cursor Down One

D 5
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 77
(23)

LIB
V04

: 2808

3044 1 !<BLF/PAGE>

```
2810 3045 1 %SBTTL 'TRIMMED_LENGTH - Calc. trimmed length of string'
2811 3046 1 ROUTINE TRIMMED_LENGTH (
2812 3047 1     LENGTH : REF VECTOR [, WORD],
2813 3048 1     ADDR
2814 3049 1 ) =
2815 3050 1 ++
2816 3051 1 FUNCTIONAL DESCRIPTION:
2817 3052 1
2818 3053 1     Calculates length of specified string ignoring trailing
2819 3054 1     blanks.
2820 3055 1
2821 3056 1 CALLING SEQUENCE:
2822 3057 1
2823 3058 1     NEW_LENGTH.wlc.v = TRIMMED_LENGTH ( LENGTH.rw.r, ADDR.ra.r )
2824 3059 1
2825 3060 1 FORMAL PARAMETERS:
2826 3061 1
2827 3062 1     LENGTH.rw.r     Address of original length of string.
2828 3063 1
2829 3064 1     ADDR.ra.r       Address of start of string.
2830 3065 1
2831 3066 1 IMPLICIT INPUTS:
2832 3067 1
2833 3068 1     NONE
2834 3069 1
2835 3070 1 IMPLICIT OUTPUTS:
2836 3071 1
2837 3072 1     NONE
2838 3073 1
2839 3074 1 ROUTINE VALUE:
2840 3075 1
2841 3076 1     NEW_LENGTH.wl.v   Length of string ignoring trailing blanks.
2842 3077 1
2843 3078 1 SIDE EFFECTS:
2844 3079 1
2845 3080 1     NONE
2846 3081 1 --
2847 3082 1
2848 3083 2 BEGIN
2849 3084 2
2850 3085 2 LOCAL
2851 3086 2     LEN_TO_SEARCH,      ! Current length to search
2852 3087 2     ADDR_TO_SEARCH ;    ! Current starting position to search
2853 3088 2
2854 3089 2 ++
2855 3090 2 Initialize local length-to-search variable and quit immediately
2856 3091 2 if we're looking at a zero-length string.
2857 3092 2 --
2858 3093 2 IF (LEN_TO_SEARCH = .LENGTH[0]) LEQU 0 THEN RETURN 0 ;
2859 3094 2
2860 3095 2 ADDR_TO_SEARCH = .ADDR ;
2861 3096 2
2862 3097 2 WHILE 1 DO
2863 3098 3 BEGIN
2864 3099 3     LOCAL
2865 3100 3     FIRST_BLANK_POS,    ! Addr. of 1st blank
2866 3101 3     NON_BLANK_POS,     ! Addr. of 1st non-blank
```


PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
TRIMMED_LENGTH - Calc. trimmed length of string

G 5
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 80
(24)

	50	04	05	12	00017	BNEQ	3\$		3113
			BC	3C	00019	MOVZWL	@LENGTH, R0		
				04	0001D	RET			
50	52		54	C3	0001E	SUBL3	FIRST_BLANK_POS, ADDR_TO_SEARCH, R0		3119
53	50		55	C1	00022	ADDL3	LEN_TO_SEARCH, R0, NEW_LEN		
64	53		20	3B	00026	SKPC	#32, NEW_LEN, (FIRST_BLANK_POS)		3127
			02	12	0002A	BNEQ	4\$		
			51	D4	0002C	CLRL	R1		
	50		51	D0	0002E	MOVL	R1, NON_BLANK_POS		
			0B	12	00031	BNEQ	5\$		3130
	51	04	BC	3C	00033	MOVZWL	@LENGTH, R1		3132
	51		53	C2	00037	SUBL2	NEW_LEN, R1		
	50		51	D0	0003A	MOVL	R1, R0		
				04	0003D	RET			
51	54		50	C3	0003E	SUBL3	NON_BLANK_POS, FIRST_BLANK_POS, R1		3136
55	51		53	C1	00042	ADDL3	NEW_LEN, R1, LEN_TO_SEARCH		
	52		50	D0	00046	MOVL	NON_BLANK_POS, ADDR_TO_SEARCH		3138
			C1	11	00049	BRB	1\$		3097
			50	D4	0004B	CLRL	R0		3144
			04	0004D	RET				

; Routine Size: 78 bytes, Routine Base: _LIB\$CODE + 09E3

; 2910 3145 1 !<BLF/PAGE>

```
: 2912 3146 1 %SBTTL 'SCR$$GET_TYPE - Get device type'
: 2913 3147 1 GLOBAL ROUTINE SCR$$GET_TYPE_R3 (
: 2914 3148 1     REQUEST TYPE;           ! Request type
: 2915 3149 1     TCB_ADDR : REF VECTOR [,LONG], ! Addr of terminal control block
: 2916 3150 1     DEV_TYPE : REF VECTOR [,LONG]   ! Device type
: 2917 3151 1 ) : GET_TYPE_LINK =
: 2918 3152 1
: 2919 3153 1 ++
: 2920 3154 1   FUNCTIONAL DESCRIPTION:
: 2921 3155 1       This routine is called by other screen package routines to
: 2922 3156 1       find the terminal type using the GETDVI system service. This
: 2923 3157 1       is done only once per terminal for the life of the process --
: 2924 3158 1       all later calls simply return the saved terminal type.
: 2925 3159 1
: 2926 3160 1   CALLING SEQUENCE:
: 2927 3161 1
: 2928 3162 1       ret_status.wlc.v = SCR$$GET_TYPE_R3 ( REQ_TYPE.rl.v; TCB.wa.r, DEV_TYPE.wl.r )
: 2929 3163 1
: 2930 3164 1   FORMAL PARAMETERS:
: 2931 3165 1
: 2932 3166 1       REQ_TYPE.rl.v   Request type.
: 2933 3167 1
: 2934 3168 1       TCB.wa.r        Address of terminal control block
: 2935 3169 1
: 2936 3170 1       DEV_TYPE.wl.r   Device type.
: 2937 3171 1
: 2938 3172 1   IMPLICIT INPUTS:
: 2939 3173 1
: 2940 3174 1       NONE
: 2941 3175 1
: 2942 3176 1   IMPLICIT OUTPUTS:
: 2943 3177 1
: 2944 3178 1       NONE
: 2945 3179 1
: 2946 3180 1   COMPLETION STATUS:
: 2947 3181 1
: 2948 3182 1       LIB$NO_STRUCT
: 2949 3183 1       statuses from SCR$SET_OUTPUT or SCR$$FOREIGN
: 2950 3184 1
: 2951 3185 1   SIDE EFFECTS:
: 2952 3186 1
: 2953 3187 1       NONE
: 2954 3188 1   --
: 2955 3189 1
: 2956 3190 2   BEGIN
: 2957 3191 2   LOCAL
: 2958 3192 2       STATUS,
: 2959 3193 2       TCB;
: 2960 3194 2
: 2961 3195 2   BUILTIN
: 2962 3196 2   AP;
: 2963 3197 2
: 2964 3198 2   MAP
: 2965 3199 2       TCB: REF BLOCK [,BYTE];
: 2966 3200 2
: 2967 3201 2   IF .SCR$LCUROUTPUT EQL 0
: 2968 3202 2   THEN
```

```
2969 3203 3 BEGIN ! no output device active
2970 3204 3 IF .SCR$L_FLINKHEAD NEQ 0
2971 3205 3 THEN
2972 3206 3 RETURN (LIB$ NO STRACT); ! error if not 1st call to scrpkg
2973 3207 3 STATUS = SCR$SET_OUTPUT (0); ! set up stream 0 for output
2974 3208 3 IF NOT .STATUS
2975 3209 3 THEN
2976 3210 3 RETURN (.STATUS);
2977 3211 2 END;
2978 3212 2
2979 3213 2 TCB = .SCR$L_CUROUTPUT; ! should now be a current ctrl block
2980 3214 2 TCB_ADDR = .TCB; ! ret addr of TCB to user
2981 3215 2
2982 3216 2 IF (DEV_TYPE = .TCB [SCR$B_TYPE]) EQL VTFORIGN
2983 3217 2 THEN
2984 3218 3 BEGIN ! Foreign processing
2985 3219 3 DEV_TYPE = .TCB [SCR$B_DEVTYPE];
2986 3220 3 IF .REQUEST_TYPE GEQ 0 ! special processing if foreign device
2987 3221 3 THEN
2988 3222 4 BEGIN
2989 3223 5 IF NOT (STATUS = SCR$$FOREIGN (.TCB, .REQUEST_TYPE,
2990 3224 5 .TCB [SCR$B_DEVTYPE], .AP))
2991 3225 5 ! note that AP still points to the argument
2992 3226 5 ! list from the calling SCR$ routine
2993 3227 4 THEN
2994 3228 4 RETURN (.STATUS);
2995 3229 3 END;
2996 3230 2 END; ! Foreign processing
2997 3231 2
2998 3232 2 RETURN (SS$_NORMAL);
2999 3233 2
3000 3234 1 END; ! end of SCR$$GET_TYPE
```

```
50 DD 00000 SCR$$GET TYPE_R3::
00000000' EF D5 00002 PUSHL R0 3147
1D 12 00008 TSTL SCR$L_CUROUTPUT 3201
00000000' EF D5 0000A BNEQ 2$
09 13 00010 TSTL SCR$L_FLINKHEAD 3204
50 00000000G 8F D0 00012 BEQL 1$
3D 11 00019 MOVL #LIB$ NO_STRACT, R0 3206
7E D4 0001B 1$: BRB 4$
00000000G 00 01 FB 0001D CLRL -(SP) 3207
31 50 E9 00024 CALLS #1, SCR$SET_OUTPUT
51 00000000' EF D0 00027 2$: BLBC STATUS, 4$ 3208
53 51 D0 0002E MOVL SCR$L_CUROUTPUT, TCB 3213
52 0A A1 9A 00031 MOVL TCB, TCB_ADDR 3214
04 52 D1 00035 MOVZBL 10(TCB), DEV_TYPE 3216
52 0B A1 9A 0003A CMPL DEV_TYPE, #4-
6E D5 0003E BNEQ 3$
13 19 00040 MOVZBL 11(TCB), DEV_TYPE 3219
5C DD 00042 TSTL REQUEST_TYPE 3220
BLSS 3$
PUSHL AP 3224
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SCR\$\$GET_TYPE - Get device type

J 5
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 83
(25)

7E	0B	A1	9A	00044	MOVZBL	11(TCB), -(SP)
	08	AE	DD	00048	PUSHL	REQUEST_TYPE
		51	DD	0004B	PUSHL	TCB
0000V	CF	04	FB	0004D	CALLS	#4, SCR\$\$FOREIGN
	03	50	E9	00052	BLBC	STATUS, 4\$
	50	01	D0	00055	MOVL	#1, R0
	51	53	D0	00058	MOVL	R3, R1
	5E	04	C0	0005B	ADDL2	#4, SP
		05	0005E		RSB	

: 3223

: 3232

: 3234

; Routine Size: 95 bytes, Routine Base: _LIB\$CODE + 0A31

```
3002 3235 1 %SBTTL 'SCR$$FOREIGN - Call foreign terminal handler'
3003 3236 1 GLOBAL ROUTINE SCR$$FOREIGN (
3004 3237 1     TCB : REF BLOCK [, BYTE], ! Addr of terminal control block
3005 3238 1     REQ_TYPE, ! Request type
3006 3239 1     DEV_TYPE, ! Device type
3007 3240 1     SCR_ARGS, ! Addr of SCR$ arguments
3008 3241 1 ) =
3009 3242 1 --
3010 3243 1 ++
3011 3244 1 FUNCTIONAL DESCRIPTION:
3012 3245 1     This routine attempts to map the foreign terminal handler into
3013 3246 1     P0 region and call it.
3014 3247 1
3015 3248 1 CALLING SEQUENCE:
3016 3249 1
3017 3250 1     ret_status.wlc.v = SCR$$FOREIGN ( TCB.ra.v, REQ_TYPE.rl.v,
3018 3251 1     DEV_TYPE.rl.v, SCR_ARGS.ra.v )
3019 3252 1
3020 3253 1 FORMAL PARAMETERS:
3021 3254 1
3022 3255 1     TCB.ra.v      Address of terminal control block
3023 3256 1
3024 3257 1     REQ_TYPE.rl.v Request type.
3025 3258 1
3026 3259 1     DEV_TYPE.rl.v Device type.
3027 3260 1
3028 3261 1     SCR_ARGS.ra.v Address of arguments from SCR$ routine
3029 3262 1
3030 3263 1 IMPLICIT INPUTS:
3031 3264 1
3032 3265 1     NONE
3033 3266 1
3034 3267 1 IMPLICIT OUTPUTS:
3035 3268 1
3036 3269 1     NONE
3037 3270 1
3038 3271 1 COMPLETION STATUS:
3039 3272 1
3040 3273 1     Statuses returned by LIB$CALL_IMAGE and OUTPUT.
3041 3274 1
3042 3275 1 SIDE EFFECTS:
3043 3276 1
3044 3277 1     NONE
3045 3278 1 --
3046 3279 1
3047 3280 2 BEGIN
3048 3281 2
3049 3282 2 BUILTIN
3050 3283 2 CALLG ;
3051 3284 2
3052 3285 2 LOCAL
3053 3286 2
3054 3287 2     DEFSTR : VECTOR [1, 8] INITIAL (                ! Default name
3055 3288 2         DSC$K_DTYPE ^24 + DSC$K_CLASS_S ^16 + 16, ! string
3056 3289 2         UPLIT('SYS$LIBRARY:.EXE'),
3057 3290 2
3058 3291 2     SCRFT : VECTOR [1, 8] INITIAL (                ! File name
```

```
3059 3292 2          DSC$K_DTYPE T ^24 + DSC$K_CLASS_S ^16 + 5, ! string
3060 3293 2          UPLIT('SCRFT')) ,
3061 3294 2
3062 3295 2          RET_STATUS,                ! Status of subroutine
3063 3296 2                                     ! calls.
3064 3297 2
3065 3298 2          LOC_BUF_DESCR : BLOCK [8, BYTE],    ! Fixed string descr.
3066 3299 2                                     ! of local buffer
3067 3300 2
3068 3301 2          OUT_LENGTH,                ! Length of output
3069 3302 2                                     ! returned in
3070 3303 2                                     ! LOCAL_BUFFER
3071 3304 2
3072 3305 2          PARM_LIST : VECTOR [15, LONG],    ! Parameter list for
3073 3306 2                                     ! call to LIB$CALL_IMAGE
3074 3307 2
3075 3308 2          STATUS;
3076 3309 2
3077 3310 2          MAP SCRFT : BLOCK [8, BYTE] ; ! To allow accessing as a descriptor
3078 3311 2
3079 3312 2          MAP DEFSTR : BLOCK [8, BYTE] ; ! To allow accessing as a descriptor
3080 3313 2
3081 3314 2      + Initialize LOC_BUF_DESCR
3082 3315 2      -
3083 3316 2          LOC_BUF_DESCR [DSC$W_LENGTH] = .TCB [SCR$L_BUFSIZ] ;
3084 3317 2          LOC_BUF_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S ;
3085 3318 2          LOC_BUF_DESCR [DSC$B_DTYPE] = DSC$K_DTYPE_T ;
3086 3319 2
3087 3320 2
3088 3321 2          STATUS = LIB$GET_VM (TCB [SCR$L_BUFSIZ], LOC_BUF_DESCR [DSC$A_POINTER]);
3089 3322 2          IF NOT .STATUS THEN RETURN (.STATUS);
3090 3323 2
3091 3324 2      + Set up elements of PARM_LIST in preparation for call to LIB$CALL_IMAGE
3092 3325 2      -
3093 3326 2          PARM_LIST [0] = 14 ; ! Number of arguments
3094 3327 2
3095 3328 2          PARM_LIST [1] = PARM_LIST [4] ; ! Address of SCRFT descr in
3096 3329 2                                     ! call list
3097 3330 2
3098 3331 2          PARM_LIST [2] = PARM_LIST [6] ; ! Address of DEFSTR descr in
3099 3332 2                                     ! call list
3100 3333 2
3101 3334 2
3102 3335 2          PARM_LIST [3] = PARM_LIST [8] ; ! Address of sub-call list
3103 3336 2
3104 3337 2          PARM_LIST [4] = .SCRFT [DSC$W_LENGTH] ; ! Length field of SCRFT
3105 3338 2
3106 3339 2          PARM_LIST [5] = .SCRFT [DSC$A_POINTER] ; ! Address field of SCRFT
3107 3340 2
3108 3341 2          PARM_LIST [6] = .DEFSTR [DSC$W_LENGTH] ; ! Length field of DEFSTR
3109 3342 2
3110 3343 2          PARM_LIST [7] = .DEFSTR [DSC$A_POINTER] ; ! Address field of DEFSTR
3111 3344 2
3112 3345 2          PARM_LIST [8] = 5 ; ! Number of arguments
3113 3346 2
3114 3347 2          PARM_LIST [9] = .SCR_ARGS ; ! Address of args from SCR$ routine
3115 3348 2
```

```
3116 3349 2  PARM_LIST [10] = .REQ_TYPE ; . Request type
3117 3350 2
3118 3351 2  PARM_LIST [11] = .DEV_TYPE - (DTS_FT1 - 1) ; ! Foreign terminal
3119 3352 2                                     ! number
3120 3353 2
3121 3354 2  PARM_LIST [12] = LOC_BUF_DESCR ; ! Address of LOC_BUF_DESC
3122 3355 2
3123 3356 2  PARM_LIST [13] = OUT_LENGTH ; ! Address of OUT_LENGTH
3124 3357 2
3125 3358 2  PARM_LIST [14] = .TCB [SCR$L_FTDATA];
3126 3359 2
3127 3360 2  +
3128 3361 2  Call LIB$CALL_IMAGE. If we get don't get a success return, quit
3129 3362 2  immediately.
3130 3363 2  -
3131 3364 2  IF NOT ( RET_STATUS = CALLG ( PARM_LIST, LIB$CALL_IMAGE))
3132 3365 2  THEN
3133 3366 2  RETURN ( .RET_STATUS ) ;
3134 3367 2
3135 3368 2  +
3136 3369 2  Call OUTPUT with the buffer returned.
3137 3370 2  -
3138 3371 2  LOC_BUF_DESCR [DSC$W_LENGTH] = .OUT_LENGTH ; ! Adjust length
3139 3372 2
3140 3373 2  RET_STATUS = OUTPUT ( .TCB, LOC_BUF_DESCR ) ;
3141 3374 2
3142 3375 2  +
3143 3376 2  Free buffer space before exiting
3144 3377 2  -
3145 3378 2
3146 3379 2  STATUS = LIB$FREE_VM (TCB [SCR$L_BUFSIZ], LOC_BUF_DESCR [DSC$A_POINTER]);
3147 3380 2  IF NOT .STATUS THEN RETURN (.STATUS);
3148 3381 2
3149 3382 2  RETURN (.RET_STATUS);
3150 3383 1  END; ! End of routine SCR$$FOREIGN
```

```
58 45 2E 3A 59 52 41 52 42 49 4C 24 53 59 53 00A90 P.AAS: .ASCII \SYSS$LIBRARY:.EXE\
00 00 00 54 46 52 43 53 45 00A9F
00AA0 P.AAT: .ASCII \SCRFT\<0><0><0>
```

```
000C 00000 .ENTRY SCR$$FOREIGN, Save R2,R3
50 5E A8 AE 9E 00002 MOVAB -88(SP), SP
54 AE 0E010010 8F D0 00006 MOVL #234946576, DEFSTR
48 AE 0E010005 8F D0 0000E MOVAB P.AAS, DEFSTR+4
4C AE DA AF 9E 0001B MOVL #234946565, SCRFT
52 04 AC D0 00020 MOVAB P.AAT, SCRFT+4
40 AE 4C A2 B0 00024 MOVL TCB, R2
42 AE 010E 8F B0 00029 MOVW 76(R2), LOC_BUF_DESCR
44 AE 9F 0002F PUSHAB #270, LOC_BUF_DESCR+2
4C A2 9F 00032 PUSHAB LOC_BUF_DESCR+4
00000000G 00 02 FB 00035 CALLS #2, LIB$GET_VM
53 50 D0 0003C MOVL R0, STATUS
```

```
3236
3280
3317
3319
3321
```


		03		53	E8	0003F	BLBS	STATUS, 1\$	3322	
				0082	31	00042	BRW	2\$		
	04	AE		0E	D0	00045	1\$:	MOVL	#14, PARM_LIST	3327
	08	AE	14	AE	9E	00049	MOVAB	PARM_LIST+16, PARM_LIST+4	3329	
	0C	AE	1C	AE	9E	0004E	MOVAB	PARM_LIST+24, PARM_LIST+8	3332	
	10	AE	24	AE	9E	00053	MOVAB	PARM_LIST+32, PARM_LIST+12	3335	
	14	AE	48	AE	3C	00058	MOVZWL	SCRFT, PARM_LIST+18	3337	
	18	AE	4C	AE	D0	0005D	MOVL	SCRFT+4, PARM_LIST+20	3339	
	1C	AE	50	AE	3C	00062	MOVZWL	DEFSTR, PARM_LIST+24	3341	
	20	AE	54	AE	D0	00067	MOVL	DEFSTR+4, PARM_LIST+28	3343	
	24	AE		05	D0	0006C	MOVL	#5, PARM_LIST+32	3345	
	28	AE	10	AC	D0	00070	MOVL	SCR_ARGS, PARM_LIST+36	3347	
	2C	AE	08	AC	D0	00075	MOVL	REQ_TYPE, PARM_LIST+40	3349	
30	AE	0C		0F	C3	0007A	SUBL3	#15, DEV_TYPE, PARM_LIST+44	3351	
		34	AE	40	AE	9E	00080	MOVAB	LOC_BUF_DESCR, PARM_LIST+48	3354
		38	AE		6E	9E	00085	MOVAB	OUT_LENGTH, PARM_LIST+52	3356
		3C	AE	50	A2	D0	00089	MOVL	80(R2), PARM_LIST+56	3358
	00000000G	00		04	AE	FA	0008E	CALLG	PARM_LIST, LIB\$CALL_IMAGE	3364
		52		50	D0	00096	MOVL	R0, RET_STATUS		
		2F		50	E9	00099	BLBC	R0, 3\$		
	40	AE		6E	B0	0009C	MOVW	OUT_LENGTH, LOC_BUF_DESCR	3371	
			40	AE	9F	000A0	PUSHAB	LOC_BUF_DESCR	3373	
			04	AC	DD	000A3	PUSHL	TCB		
	0000V	CF		02	FB	000A6	CALLS	#2, OUTPUT		
		52		50	D0	000AB	MOVL	R0, RET_STATUS		
			44	AE	9F	000AE	PUSHAB	LOC_BUF_DESCR+4	3379	
7E	04	AC	0000004C	8F	C1	000B1	ADDL3	#76, TCB, -(SP)		
	00000000G	00		02	FB	000BA	CALLS	#2, LIB\$FREE_VM		
		53		50	D0	000C1	MOVL	R0, STATUS		
		04		53	E8	000C4	BLBS	STATUS, 3\$	3380	
		50		53	D0	000C7	2\$:	MOVL	STATUS, R0	
				04	000CA		RET			
		50		52	D0	000CB	3\$:	MOVL	RET_STATUS, R0	3382
				04	000CE		RET			3383

; Routine Size. 207 bytes, Routine Base: _LIB\$CODE + 0AA8

; 3151 3384 1 !<BLF/PAGE>

```
3153 3385 1 XSBTTL 'OUTPUT - Write string to terminal without carriage control'
3154 3386 1 ROUTINE OUTPUT ( TCB : REF BLOCK [,BYTE],
3155 3387 1 STRING : REF BLOCK [,BYTE] ) =
3156 3388 1 ++
3157 3389 1 FUNCTIONAL DESCRIPTION:
3158 3390 1 This routine writes the specified string to the terminal
3159 3391 1 without carriage control.
3160 3392 1
3161 3393 1 CALLING SEQUENCE:
3162 3394 1 ret_status.wlc.v = OUTPUT ( TCB.rab.r, string.rt.ds )
3163 3395 1
3164 3396 1 FORMAL PARAMETERS:
3165 3397 1
3166 3398 1 TCB.rab.r Address of current terminal control block.
3167 3399 1
3168 3400 1 STRING.rt.ds Address of fixed string descriptor of output
3169 3401 1 string.
3170 3402 1
3171 3403 1 IMPLICIT INPUTS:
3172 3404 1
3173 3405 1 NONE
3174 3406 1
3175 3407 1 IMPLICIT OUTPUTS:
3176 3408 1
3177 3409 1 NONE
3178 3410 1
3179 3411 1 ROUTINE STATUS:
3180 3412 1
3181 3413 1 SIDE EFFECTS:
3182 3414 1
3183 3415 1 String will be output.
3184 3416 1
3185 3417 1 --
3186 3418 1
3187 3419 1 BEGIN
3188 3420 1
3189 3421 2 LOCAL
3190 3422 2 STATUS : INITIAL (SS$_NORMAL) ; ! Status to return to caller
3191 3423 2
3192 3424 2 If input string length is 0, return an immediate success.
3193 3425 2
3194 3426 2 IF .STRING [DSC$W_LENGTH] EQL 0 THEN RETURN .STATUS ;
3195 3427 2
3196 3428 2
3197 3429 2 Major decision is whether output is being buffered or not.
3198 3430 2
3199 3431 2 IF .TCB [SCR$L_BUFFER] EQL 0 ! If output being buffered
3200 3432 2 THEN
3201 3433 2 BEGIN ! Not buffered
3202 3434 2 IF .TCB [SCR$L_RTNADDR] EQL 0 ! If user has supplied call-back
3203 3435 2 THEN ! routine
3204 3436 3 BEGIN ! No call-back routine
3205 3437 3 LOCAL
3206 3438 3 CHANNEL ; ! Channel as a longword
3207 3439 4
3208 3440 4
3209 3441 4
```

```
: 3210      3442  4
: 3211      3443  4      CHANNEL = .TCB [SCR$W_CHAN] ;
: 3212      3444  4
: 3213      3445  4      IF .TCB [SCR$B_TYPE] EQL 0 ! If unknown type
: 3214      3446  4      THEN
: 3215      3447  5          BEGIN ! Use RMS OUTPUT
: 3216      3448  5          STATUS = RMS_OUTPUT ( .TCB, .STRING ) ;
: 3217      3449  5          END ! Use RMS_OUTPUT
: 3218      3450  4      ELSE
: 3219      3451  5          BEGIN ! Use QIO
: 3220      3452  5          STATUS = $QIO ( CHAN = .CHANNEL,
: 3221      3453  5          EFN = .TCB [SCR$L_EFN],
: 3222      3454  5          FUNC = (IOS$WRITEVBLK OR IOS$M_NOFORMAT),
: 3223      3455  5          P1 = .STRING [DSC$A_POINTER],
: 3224      3456  5          P2 = .STRING [DSC$W_LENGTH] ) ;
: 3225      3457  4          END ; ! Use QIO
: 3226      3458  4      END ! No call-back routine
: 3227      3459  4
: 3228      3460  3      ELSE
: 3229      3461  3
: 3230      3462  4          BEGIN ! Call-back routine provided
: 3231      3463  4          STATUS = RMS_OUTPUT ( .TCB, .STRING ) ;
: 3232      3464  3          END ! Call-back routine provided
: 3233      3465  3      END ! Not buffered
: 3234      3466  3
: 3235      3467  2      ELSE
: 3236      3468  2
: 3237      3469  3          BEGIN ! Buffered output
: 3238      3470  3          !+
: 3239      3471  3          ! Description of what the buffer header (BH) looks like:
: 3240      3472  3          !-
: 3241      3473  3          MACRO
: 3242      3474  3              BH$W_CURR_LENGTH = 0, 0, 16, 0 % ; ! Current no. of bytes
: 3243      3475  3              BH$A_POINTER = 4, 0, 32, 0 % ; ! Addr of 1st byte
: 3244      3476  3              BH$W_MAX_LENGTH = 8, 0, 16, 0 % ; ! Max. no. of bytes
: 3245      3477  3
: 3246      3478  3          BIND BH = .TCB [SCR$L_BUFFER] ;
: 3247      3479  3          MAP BH : BLOCK [,BYTE] ;
: 3248      3480  3
: 3249      3481  3          LOCAL
: 3250      3482  3              BYTES_REMAINING,
: 3251      3483  3              SAVED_BUFFER_ADDR ;
: 3252      3484  3
: 3253      3485  3          SAVED_BUFFER_ADDR = .TCB [SCR$L_BUFFER] ;
: 3254      3486  3          BYTES_REMAINING = .BH [BH$W_MAX_LENGTH] -
: 3255      3487  3              .BH [BH$W_CURR_LENGTH] ;
: 3256      3488  3
: 3257      3489  3          IF .STRING [DSC$W_LENGTH] GTRU .BYTES_REMAINING
: 3258      3490  3          THEN
: 3259      3491  4              BEGIN ! Won't fit in current buffer
: 3260      3492  4              !+
: 3261      3493  4              ! Check to see if length of current string is larger than
: 3262      3494  4              ! the entire size of the buffer.
: 3263      3495  4              !-
: 3264      3496  4              IF .STRING [DSC$W_LENGTH] GTRU .BH [BH$W_MAX_LENGTH]
: 3265      3497  4              THEN
: 3266      3498  5                  BEGIN ! Will never fit
```

```

: 3267      3499 5      STATUS = LIB$ SCRBUFOVF ;
: 3268      3500 5      END      ! Will never fit
: 3269      3501 4      ELSE
: 3270      3502 5      BEGIN      ! Flush buffer, then add to buffer
: 3271      3503 5      SCR$PUT BUFFER (0) ;
: 3272      3504 5      TCB [SCR$SL_BUFFER] = .SAVED_BUFFER_ADDR;
: 3273      3505 5      ! this was re-init'd by SCR$PUT_BUFFER
: 3274      3506 5      CH$MOVE ( .STRING [DSC$W_LENGTH],
: 3275      3507 5      .STRING [DSC$A_POINTER],
: 3276      3508 5      .BH [BH$A_POINTER] + .BH [BH$W_CURR_LENGTH] ) ;
: 3277      3509 5
: 3278      3510 5      BH [BH$W_CURR_LENGTH] = .BH [BH$W_CURR_LENGTH] +
: 3279      3511 5      .STRING [DSC$W_LENGTH] ;
: 3280      3512 4      END ;      ! Flush buffer, then add to buffer
: 3281      3513 4      END      ! Won't fit in current buffer
: 3282      3514 3      ELSE
: 3283      3515 4      BEGIN      ! Will fit in buffer
: 3284      3516 4      CH$MOVE ( .STRING [DSC$W_LENGTH],
: 3285      3517 4      .STRING [DSC$A_POINTER],
: 3286      3518 4      .BH [BH$A_POINTER] + .BH [BH$W_CURR_LENGTH] ) ;
: 3287      3519 4
: 3288      3520 4      BH [BH$W_CURR_LENGTH] = .BH [BH$W_CURR_LENGTH] +
: 3289      3521 4      .STRING [DSC$W_LENGTH] ;
: 3290      3522 3      END ;      ! Will fit in buffer
: 3291      3523 2      END ;      ! Buffered output
: 3292      3524 2
: 3293      3525 2      RETURN (.STATUS) ;
: 3294      3526 2
: 3295      3527 1      END;      ! End of routine OUTPUT
```

```

                                01FC 00000 OUTPUT: .WORD      Save R2,R3,R4,R5,R6,R7,R8
58      01  D0 00002      MOVL      #1, STATUS
57      08  AC  D0 00005      MOVL      STRING, R7
        67  B5 00009      TSTW      (R7)
        68  13 0000B      BEQL      4$
52      04  AC  D0 0000D      MOVL      TCB, R2
56      04  A2  D0 00011      MOVL      4(R2), R6
        3D  12 00015      BNEQ      3$
        38  D5 00017      TSTL      56(R2)
        2A  12 0001A      BNEQ      1$
53      08  A2  3C 0001C      MOVZWL   8(R2), CHANNEL
        0A  A2  95 00020      TSTB      10(R2)
        21  13 00023      BEQL      1$
        7E  7C 00025      CLRQ      -(SP)
        7E  7C 00027      CLRQ      -(SP)
7E      04  67  3C 00029      MOVZWL   (R7), -(SP)
        7E  7C 0002F      CLRQ      -(SP)
        7E  D4 00031      CLRL      -(SP)
7E      0130 8F  3C 00033      MOVZWL   #304, -(SP)
        53  DD 0C038      PUSHL      CHANNEL
        48  A2  DD 0003A      PUSHL      72(R2)
00000000G 00      0C  FB 0003D      CALLS      #12, SYSSQ10W
```

```

: 3386
: 3421
: 3429
:
: 3434
:
: 3437
:
: 3443
: 3445
: 3456
:
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management

OUTPUT - Write string to terminal without carri

E 6
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;i

Page 91
(27)

LI
VO

50	67	0000V	CF	0084	09 11 00044	BRB	2\$		
			58		8F BB 00046 1\$:	PUSHR	#^M<R2,R7>		3463
					02 FB 0004A	CALLS	#2, RMS_OUTPUT		
			53		50 D0 0004F 2\$:	MOVL	R0, STATUS		
			50		3D 11 00052	BRB	7\$		3434
			51	08	56 D0 00054 3\$:	MOVL	R6, SAVED_BUFFER_ADDR		3485
			50		A6 3C 00057	MOVZWL	8(R6), BYTES_REMAINING		3487
			51		66 3C 00058	MOVZWL	(R6), R1		
			50		51 C2 0005E	SUBL2	R1, BYTES_REMAINING		
			10		00 ED 00061	CMPZV	#0, #16, (R7), BYTES_REMAINING		3489
					1A 1B 00066	BLEQU	6\$		
		08	A6		67 B1 00068	CMPW	(R7), 8(R6)		3496
					09 1B 0006C	BLEQU	5\$		
			58	00000000G	8F D0 0006E	MOVL	#LIB\$_SCRBUFOVF, STATUS		3499
					1A 11 00075 4\$:	BRB	7\$		3496
					7E D4 00077 5\$:	CLRL	-(SP)		3503
		F873	CF		01 FB 00079	CALLS	#1, SCR\$PUT_BUFFER		
		04	A2		53 D0 0007E	MOVL	SAVED_BUFFER_ADDR, 4(R2)		3504
			50		66 3C 00082 6\$:	MOVZWL	(R6), R0		3518
			50	04	A6 C0 00085	ADDL2	4(R6), R0		
	60	04	B7		67 28 00089	MOVCL3	(R7), 24(R7), (R0)		
			66		67 A0 0008E	ADDW2	(R7), (R6)		3521
			50		58 D0 00091 7\$:	MOVL	STATUS, R0		3525
					04 00094	RET			3527

; Routine Size: 149 bytes, Routine Base: _LIB\$CODE + 0B77

; 3296 3528 1 !<BLF/PAGE>

```
3298 3529 1 %SBTTL 'FILL_MAP - Fill map with specified character'
3299 3530 1 ROUTINE FILL_MAP (
3300 3531 1     TCB : REF BLOCK [,BYTE],
3301 3532 1     START_LINE,
3302 3533 1     START_COL,
3303 3534 1     END_LINE,
3304 3535 1     END_COL,
3305 3536 1     FILE_CHAR
3306 3537 1 ) =
3307 3538 1
3308 3539 1 ++
3309 3540 1 FUNCTIONAL DESCRIPTION:
3310 3541 1     This routine fills the character map with a specified character,
3311 3542 1     zeroes the attribute map, and sets the modified map to ON.
3312 3543 1
3313 3544 1 CALLING SEQUENCE:
3314 3545 1
3315 3546 1     ret_status.wlc.v = FILL_MAP (TCB.rz.r, START_LINE.ml.v, START_COL.ml.v,
3316 3547 1     END_LINE.ml.v, END_COL.ml.v, FILE_CHAR.rt.r)
3317 3548 1
3318 3549 1 FORMAL PARAMETERS:
3319 3550 1
3320 3551 1     TCB.rz.r           Terminal Control Block
3321 3552 1     START_LINE.ml.v   starting line number
3322 3553 1     START_COL.ml.v    starting column number
3323 3554 1     END_LINE.ml.v     ending line number
3324 3555 1     END_COL.ml.v      ending column number
3325 3556 1     FILE_CHAR.rt.r    fill character
3326 3557 1
3327 3558 1 IMPLICIT INPUTS:
3328 3559 1
3329 3560 1     NONE
3330 3561 1
3331 3562 1 IMPLICIT OUTPUTS:
3332 3563 1
3333 3564 1     NONE
3334 3565 1
3335 3566 1 COMPLETION STATUS:
3336 3567 1
3337 3568 1
3338 3569 1 SIDE EFFECTS:
3339 3570 1
3340 3571 1     NONE
3341 3572 1 --
3342 3573 1
3343 3574 2 BEGIN
3344 3575 2
3345 3576 2 LOCAL
3346 3577 2     START_INDEX,
3347 3578 2     END_INDEX,
3348 3579 2     LENGTH;
3349 3580 2
3350 3581 2 ++
3351 3582 2     Convert line and column numbers to zero based.
3352 3583 2 --
3353 3584 2
3354 3585 2     START_LINE = .START_LINE - 1;
```

```
: 3355      3586 2      START_COL = .START_COL - 1;
: 3356      3587 2      END_LINE = .END_LINE - 1;
: 3357      3588 2
: 3358      3589 2      !+
: 3359      3590 2      ; Compute the start index.
: 3360      3591 2      ; -
: 3361      3592 2
: 3362      3593 2      START_INDEX = (.TCB [SCR$W_DEVWIDTH] * .START_LINE) + .START_COL;
: 3363      3594 2      IF .START_INDEX GEQU .TCB [SCR$L_AREA]
: 3364      3595 2      THEN
: 3365      3596 2          RETURN 0;                      ! error if index outside area
: 3366      3597 2
: 3367      3598 2      !+
: 3368      3599 2      ; Compute the ending index + 1.
: 3369      3600 2      ; -
: 3370      3601 2
: 3371      3602 2      END_INDEX = (.TCB [SCR$W_DEVWIDTH] * .END_LINE) + .END_COL;
: 3372      3603 2      IF .END_INDEX GTRU .TCB [SCR$L_AREA]
: 3373      3604 2      THEN
: 3374      3605 2          RETURN 0;                      ! error if index outside area
: 3375      3606 2
: 3376      3607 2      !+
: 3377      3608 2      ; Fill the character and attribute maps.
: 3378      3609 2      ; -
: 3379      3610 2
: 3380      3611 2      LENGTH = .END_INDEX - .START_INDEX;
: 3381      3612 2      IF .LENGTH LEQU 0
: 3382      3613 2      THEN
: 3383      3614 2          RETURN 0;                      ! do nothing if zero or negative length
: 3384      3615 2
: 3385      3616 2      CH$FILL (.FILL_CHAR, .LENGTH, (.TCB [SCR$L_CHARMAP] + .START_INDEX));
: 3386      3617 2      CH$FILL (%C' ', .LENGTH, (.TCB [SCR$L_ATTRMAP] + .START_INDEX));
: 3387      3618 2
: 3388      3619 2      RETURN 1;
: 3389      3620 2
: 3390      3621 1      END;                      ! End of routine FILL_MAP
```

01FC 00000 FILL_MAP:

			08	AC	D7	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	: 3530
			0C	AC	D7	00005	DECL	START_LINE	: 3585
			10	AC	D7	00008	DECL	START_COL	: 3586
			04	AC	D0	0000B	DECL	END_LINE	: 3587
	56		0C	A6	3C	0C00F	MOVL	TCB, R6	: 3593
	50		08	AC	C4	00013	MOVZWL	12(R6), R0	
57	50		0C	AC	C1	00017	MULL2	START_LINE, R0	
	50						ADDL3	START_COL, R0, START_INDEX	
	14	A6					CMPL	START_INDEX, 20(R6)	: 3594
							BGEQU	1\$	
	50		0C	A6	3C	00022	MOVZWL	12(R6), R0	: 3602
	50		10	AC	C4	00026	MULL2	END_LINE, R0	
	50		14	AC	C0	0002A	ADDL2	END_COL, END_INDEX	
	14	A6					CMPL	END_INDEX, 20(R6)	: 3603
							BGTRU	1\$	
			1B	1A	00032				

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
FILL_MAP - Fill map with specified character

H 6
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 94
(28)

LIB
V04

58	18	AC	6E	18	B647	00	2C	0003A	SUBL3	START_INDEX, END_INDEX, LENGTH	:	3611
58	20	6E	1C	B647	00	2C	00043	BEQL	1\$		:	3612
		50			01	D0	0004B	MOVC5	#0, (SP), FILL_CHAR, LENGTH, @24(R6)-		:	3616
					04	0004E		MOVC5	[START_INDEX]		:	3617
					04	0004F	1\$:	MOVL	#1, R0		:	3619
					04	00051		RET	R0		:	3621
								RET			:	

; Routine Size: 82 bytes, Routine Base: _LIB\$CODE + 0C0C

; 3391 3622 1 !<BLF/PAGE>


```
3393 3623 1 %SBTTL 'PUT_MAP - Output current map'
3394 3624 1 ROUTINE PUT_MAP (
3395 3625 1     TCB : REF BLOCK [,BYTE]
3396 3626 1     ) =
3397 3627 1 ++
3398 3628 1 FUNCTIONAL DESCRIPTION:
3399 3629 1
3400 3630 1     This routine processes the internal screen, attribute and
3401 3631 1     modified maps and calls other screen package routines to
3402 3632 1     output the result.
3403 3633 1
3404 3634 1 CALLING SEQUENCE:
3405 3635 1
3406 3636 1     ret_status.wlc.v = PUT_MAP (TCB.rab.r)
3407 3637 1
3408 3638 1 FORMAL PARAMETERS:
3409 3639 1
3410 3640 1     TCB.rab.r           Current terminal control block
3411 3641 1
3412 3642 1 IMPLICIT INPUTS:
3413 3643 1
3414 3644 1     NONE
3415 3645 1
3416 3646 1 IMPLICIT OUTPUTS:
3417 3647 1
3418 3648 1     NONE
3419 3649 1
3420 3650 1 COMPLETION STATUS:
3421 3651 1
3422 3652 1     $$$_NORMAL      Normal successful completion
3423 3653 1     Errors returned by:  RMS_OUTPUT
3424 3654 1
3425 3655 1 SIDE EFFECTS:
3426 3656 1
3427 3657 1     NONE
3428 3658 1 --
3429 3659 1
3430 3660 2 BEGIN
3431 3661 2 ++
3432 3662 2 $OUTPUT_NEXT_LINE
3433 3663 2     This macro causes the next line of text from the data buffer to
3434 3664 2     be output.  If no attribute information pertains to this line, it is
3435 3665 2     output directly.  If attribute information does pertain to this line,
3436 3666 2     DO_ATTR is called to output the line with the appropriate attributes.
3437 3667 2 --
3438 3668 2 MACRO
3439 3669 2     $OUTPUT_NEXT_LINE =
3440 3670 2     SKPC (%REF(NOLL), DW2, .CURR_PTR + .TCB [SCR$L_AREA]; REM_ATTR, FND);
3441 3671 2     IF .REM_ATTR EQL 0
3442 3672 2     THEN
3443 3673 2         BEGIN ! No attributes set for this line's worth of text
3444 3674 2         ++
3445 3675 2         ! Simply output line's worth of text as it sits in data buffer
3446 3676 2         --
3447 3677 2         LOC_DESC [DSC$W_LENGTH] = .DW ;
3448 3678 2         LOC_DESC [DSC$A_POINTER] = .CURR_PTR ;
3449 3679 2         STATUS = RMS_OUTPUT ( .TCB, LOC_DESC ) ;
```

```

3450      END      ! No attributes set for this line's worth of text
3451      ELSE
3452      BEGIN      ! Attributes are present
3453      +
3454      ! Call DO_ATTR to output a line's worth of text with the
3455      ! appropriate attributes simulated (e.g., underlining and
3456      ! bolding)
3457      -
3458      DO_ATTR ( REM_ATTR, FND, .TCB, DW, .CURR_PTR );
3459      END;      ! Attributes are present
3460      X ;      ! End of macro $OUTPUT_NEXT_LINE
3461
3462      +
3463      $OUTPUT_N_BLANK_LINES
3464      ! This macro outputs .LINES_TO_SKIP blank lines, advancing the
3465      ! current pointer (CURR_POINTER) as it goes along.
3466      -
3467      MACRO
3468      $OUTPUT_N_BLANK_LINES =
3469      INCR I FROM 1 TO .LINES_TO_SKIP
3470      DO
3471      BEGIN
3472      STATUS = RMS_OUTPUT ( .TCB, LOC_DESC );
3473      CURR_PTR = .CURR_PTR + .DW2 ;      ! Advance to next line
3474      END ;
3475      X ;      ! End of macro $OUTPUT_N_BLANK_LINES
3476
3477      +
3478      $OUTPUT_FF
3479      ! This macro output a <ff> to clear an entire page.
3480      -
3481      MACRO
3482      $OUTPUT_FF =
3483      LOC_DESC [DSC$W_LENGTH] = 1 ;
3484      LOC_DESC [DSC$A_POINTER] = UPLIT ( BYTE ( FF ) );
3485      STATUS = RMS_OUTPUT ( .TCB, LOC_DESC );
3486      X ;      ! End of macro $OUTPUT_FF
3487
3488      !
3489      ! R1 = R3 = FND = 1st non-space character
3490      ! R2 = LINES_TO_SKIP
3491      ! R4 = BYTES_REMAINING
3492      ! R7 = DW
3493      ! R8 = CURR_PTR
3494      ! R9 = SIZE
3495      ! R10= END_DATA_ADDR = start of attributes
3496      ! R11 = DW2
3497
3498      BUILTIN
3499      SKPC;
3500
3501      LOCAL
3502      STATUS,      ! Status to return to caller
3503      DW,          ! Device width
3504      DW2,         ! Copy of device width
3505      CURR_PTR,    ! Pointer to start of remaining characters in
3506      map          ! map
3507      END_DATA_ADDR, ! End of character data buffer, start of
```

```

3507 3737 2 attribute buffer
3508 3738 2 BYTES_REMAINING, ! No. of bytes remaining to be output
3509 3739 2 LINES_TO_SKIP, ! No. of blank lines to skip
3510 3740 2 REM_ATTR, ! No. of attr bytes remaining
3511 3741 2 LOC_DESC : BLOCK [8, BYTE]; ! Local fixed string descriptor
3512 3742 2 ! for output
3513 3743 2
3514 3744 2 SCRS$WORKMASK = 0 ; ! Init working attribute mask
3515 3745 2 DW = .TCB [SCRS$DEVWIDTH] ; ! Device width
3516 3746 2 DW2 = .DW ; ! Copy of device width
3517 3747 2 CURR_PTR = .TCB [SCRS$CHARMAP] ; ! Address of start of character
3518 3748 2 ! buffer
3519 3749 2 END_DATA_ADDR = .TCB[SCRS$CHARMAP] + .TCB [SCRS$AREA] ;
3520 3750 2
3521 3751 2 LOC_DESC [DSC$B_CLASS] = DSC$K_CLASS_S ;
3522 3752 2 LOC_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;
3523 3753 2
3524 3754 2 WHILE (BYTES_REMAINING = .END_DATA_ADDR - .CURR_PTR) GEQ 0
3525 3755 2 DO
3526 3756 3 BEGIN ! Overall loop
3527 3757 3 LOCAL
3528 3758 3 TEMP, ! throwaway value
3529 3759 3 FND ; ! Next byte index found by a CH$FIND_NOT_CH
3530 3760 3
3531 3761 3 !+
3532 3762 3 ! Find next displayable character
3533 3763 3 SKPC (%REF(BLANK), BYTES_REMAINING, .CURR_PTR; TEMP, FND);
3534 3764 3 LINES_TO_SKIP = (.FND - .CURR_PTR) / .DW2;
3535 3765 3 IF .LINES_TO_SKIP EQL 0
3536 3766 3 THEN
3537 3767 4 BEGIN ! No lines to skip
3538 3768 4 DW = TRIMMED_LENGTH (DW, .CURR_PTR);
3539 3769 4 IF .TCB [SCRS$ATTRMASK] EQL 0
3540 3770 4 THEN
3541 3771 5 BEGIN ! No scan for attributes
3542 3772 5 LOC_DESC [DSC$W_LENGTH] = .DW ;
3543 3773 5 LOC_DESC [DSC$A_POINTER] = .CURR_PTR ;
3544 3774 5 STATUS = RMS_OUTPUT ( .TCB, LOC_DESC ) ;
3545 3775 5 END ! No scan for attributes
3546 3776 4 ELSE
3547 3777 5 BEGIN ! Scan for attributes
3548 3778 5 $OUTPUT_NEXT_LINE ;
3549 3779 4 END; ! Scan for attributes
3550 3780 4 END ! No lines to skip
3551 3781 4
3552 3782 3 ELSE
3553 3783 3
3554 3784 4 BEGIN ! Some lines to skip
3555 3785 4 IF .FND GEQ .END_DATA_ADDR AND ! If remaining page not
3556 3786 4 .LINES_TO_SKIP NEQ 1 ! blank and if more than
3557 3787 4 THEN ! one blank line
3558 3788 5 BEGIN ! A
3559 3789 5 IF .TCB [SCRS$ATTRMASK] NEQ 0
3560 3790 5 THEN
3561 3791 6 BEGIN
3562 3792 6 DW = 0 ;
3563 3793 7 IF (FND = CH$FIND_NOT_CH (

```

```

: 3564      3794 7      .BYTES REMAINING,
: 3565      3795 7      .CURR_PTR + .TCB [SCR$L_AREA], 0))
: 3566      3796 6
: 3567      3797 6      NEQ 0
: 3568      3798 7      THEN
: 3569      3799 7      BEGIN
: 3570      3800 7      LOC_DESC [DSC$W_LENGTH] = 0;
: 3571      3801 7      LOC_DESC [DSC$A_POINTER] = 0;
: 3572      3802 7      IF .TCB [SCR$L_ATTRMASK] EQL 0
: 3573      3803 8      THEN
: 3574      3804 8      BEGIN
: 3575      3805 8      $OUTPUT_N_BLANK_LINES ;
: 3576      3806 7      END
: 3577      3807 8      ELSE
: 3578      3808 8      BEGIN
: 3579      3809 8      SKPC (%REF(NULL), DW2, .CURR_PTR + .TCB [SCR$L_AREA]:
: 3580      3810 8      REM_ATTR, FND);
: 3581      3811 8      IF .REM_ATTR NEQ 0
: 3582      3812 9      THEN
: 3583      3813 9      BEGIN
: 3584      3814 9      DO_ATTR (REM_ATTR, FND, .TCB,
: 3585      3815 9      DW, .CURR_PTR) ;
: 3586      3816 8      END
: 3587      3817 9      ELSE
: 3588      3818 9      BEGIN
: 3589      3819 8      $OUTPUT_N_BLANK_LINES ;
: 3590      3820 7      END;
: 3591      3821 7      END
: 3592      3822 6      ELSE
: 3593      3823 7      BEGIN
: 3594      3824 7      $OUTPUT_FF ;
: 3595      3825 7      EXITLOOP ;
: 3596      3826 6      END;
: 3597      3827 6      END
: 3598      3828 5      ELSE
: 3599      3829 6      BEGIN
: 3600      3830 6      $OUTPUT_FF ;
: 3601      3831 6      EXITLOOP ;
: 3602      3832 5      END;
: 3603      3833 5      END ! A
: 3604      3834 4      ELSE
: 3605      3835 5      BEGIN
: 3606      3836 5      +
: 3607      3837 5      - Remainder of page is blank or next line is blank
: 3608      3838 5      -
: 3609      3839 5      LOC_DESC [DSC$W_LENGTH] = 0;
: 3610      3840 5      LOC_DESC [DSC$A_POINTER] = 0;
: 3611      3841 5      IF .TCB [SCR$L_ATTRMASK] EQL 0 ! If no attributes
: 3612      3842 5      THEN
: 3613      3843 6      BEGIN
: 3614      3844 6      $OUTPUT_N_BLANK_LINES ;
: 3615      3845 6      END
: 3616      3846 5      ELSE
: 3617      3847 6      BEGIN
: 3618      3848 6      +
: 3619      3849 6      - Locate next attribute byte
: 3620      3850 6      -
```

```
3621 3851 6 SKPC (%REF(NULL), DW2, .CURR_PTR + .TCB [SCR$L_AREA];
3622 3852 6 REM ATTR, FND);
3623 3853 6 IF .REM_ATTR NEQ 0
3624 3854 6 THEN
3625 3855 7 BEGIN
3626 3856 7 !
3627 3857 7 ! Output with attributes
3628 3858 7 !
3629 3859 7 DO_ATTR (REM_ATTR, FND, .TCB, DW,
3630 3860 7 .CURR_PTR);
3631 3861 7 END
3632 3862 6 ELSE
3633 3863 7 BEGIN
3634 3864 7 !
3635 3865 7 ! Output without attributes
3636 3866 7 !
3637 3867 7 $OUTPUT_N_BLANK_LINES;
3638 3868 6 END;
3639 3869 5 END;
3640 3870 5 DW = TRIMMED_LENGTH ( DW, .CURR_PTR);
3641 3871 5 IF .TCB [SCR$L_ATTRMASK] EQL 0
3642 3872 5 THEN
3643 3873 6 BEGIN ! No scan for attributes
3644 3874 6 LOC_DESC [DSC$W_LENGTH] = .DW;
3645 3875 6 LOC_DESC [DSC$A_POINTER] = .CURR_PTR;
3646 3876 6 STATUS = RMS_OUTPUT ( .TCB, LOC_DESC );
3647 3877 6 END ! No scan for attributes
3648 3878 5 ELSE
3649 3879 6 BEGIN ! Scan for attributes
3650 3880 6 $OUTPUT_NEXT_LINE;
3651 3881 5 END; ! Scan for attributes
3652 3882 4 END;
3653 3883 3 END; ! Some lines to skip
3654 3884 3
3655 3885 3 CURR_PTR = .CURR_PTR + .DW2; ! Advance to next line
3656 3886 3 DW = .DW2; ! Reset width to width of page
3657 3887 2 END; ! Overall loop
3658 3888 2
3659 3889 2 TCB [SCR$L_ATTRMASK] = .SCR$L_WORKMASK; ! Reset attribute mask
3660 3890 2 RETURN ( .STATUS );
3661 3891 1 END; ! End of routine PUT_MAP
```

```
00C5E .BLKB 2
OC 00C60 P.AAU: .BYTE 12
00C61 .BLKB 3
OC 00C64 P.AAV: .BYTE 12
```

```
SE 00000000' 10 C2 00002 PUT_MAP: .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 3624
53 04 AC D0 0000B SUBL2 #16, SP : 3744
7E OC A3 3C 0000F CLRL SCR$L_WORKMASK : 3745
56 6E D0 00013 MOVL TCB, R3 :
MOVL 12(R3), DW :
MOVL DW, DW2 : 3746
```

		52	18	A3	D0	00016	MOVL	24(R3), CURR_PTR	3747
		54	14	A3	9E	0001A	MOVAB	20(R3), R4	3749
5A	18	A3		64	C1	0001E	ADDL3	(R4), 24(R3), END_DATA_ADDR	
	0E	AE	010E	8F	B0	00023	MOVW	#270, LOC_DESC+2	3752
		57	2C	A3	9E	00029	MOVAB	44(R3), R7	3769
59		5A		52	C3	0002D	SUBL3	CURR_PTR, END_DATA_ADDR, BYTES_REMAINING	3754
				03	18	00031	BGEQ	2\$	
			016E	31	00033		BRW	28\$	
62		59		20	3B	00036	SKPC	#32, BYTES_REMAINING, (CURR_PTR)	3763
	04	AF		51	D0	0003A	MOVL	R1, FND	
50	04	AE		52	C3	0003E	SUBL3	CURR_PTR, FND, R0	3764
58		50		56	C7	00043	DIVL3	DW2, R0, LINES_TO_SKIP	
				03	12	00047	BNEQ	3\$	3765
			0102	31	00049		BRW	24\$	
		5A	04	AE	D1	0004C	CMPL	FND, END_DATA_ADDR	3785
				03	18	00050	BGEQ	5\$	
			0096	31	00052		BRW	17\$	
		01		58	D1	00055	CMPL	LINES_TO_SKIP, #1	3786
				F8	13	00058	BEQL	4\$	
				67	D5	0005A	TSTL	(R7)	3789
				73	13	0005C	BEQL	15\$	
00 B442		59		6E	D4	0005E	CLRL	DW	3792
				00	3B	00060	SKPC	#0, BYTES_REMAINING, @0(R4)[CURR_PTR]	3793
				02	12	00066	BNEQ	6\$	
				51	D4	00068	CLRL	R1	
	04	AE		51	D0	0006A	MOVL	R1, FND	
				55	13	0006E	BEQL	14\$	3796
			0C	AE	B4	00070	CLRW	LOC_DESC	3799
			10	AE	D4	00073	CLRL	LOC_DESC+4	3800
				67	D5	00076	TSTL	(R7)	3801
				1A	12	00078	BNEQ	9\$	
				55	D4	0007A	CLRL	I	3803
				10	11	0007C	BRB	8\$	
			0C	AE	9F	0007E	PUSHAB	LOC_DESC	
				53	DD	00081	PUSHL	R3	
	0000V	CF		02	FB	00083	CALLS	#2, RMS_OUTPUT	
		5B		50	D0	00088	MOVL	R0, STATUS	
		52		56	C0	0008B	ADDL2	DW2, CURR_PTR	
EC		55		58	F3	0008E	AOBLEQ	LINES_TO_SKIP, I, 7\$	
				2E	11	00092	BRB	13\$	3801
00 B442		56		00	3B	00094	SKPC	#0, DW2, @0(R4)[CURR_PTR]	3808
	08	AE		50	D0	0009A	MOVL	R0, REM_ATTR	
	04	AE		51	D0	0009E	MOVL	R1, FND	
			08	AE	D5	000A2	TSTL	REM_ATTR	3810
				03	13	000A5	BEQL	10\$	
			00DF	31	000A7		BRW	26\$	
				55	D4	000AA	CLRL	I	3817
				10	11	000AC	BRB	12\$	
			0C	AE	9F	000AE	PUSHAB	LOC_DESC	
				53	DD	000B1	PUSHL	R3	
	0000V	CF		02	FB	000B3	CALLS	#2, RMS_OUTPUT	
		5B		50	D0	000B8	MOVL	R0, STATUS	
		52		56	C0	000BB	ADDL2	DW2, CURR_PTR	
EC		55		58	F3	000BE	AOBLEQ	LINES_TO_SKIP, I, 11\$	
				00D6	31	000C2	BRW	27\$	3793
				01	B0	000C5	MOVW	#1, LOC_DESC	3823
	0C	AE		01	B0	000C5	MOVW	#1, LOC_DESC	
	10	AE	FF2E	CF	9E	000C9	MOVAB	P.AAU, LOC_DESC+4	

	OC	AE		0A	11	000CF	BRB	16\$		
	10	AE	FF26	01	B0	000D1	MOVW	#1, LOC_DESC	3829	
			OC	CF	9E	000D5	MOVAB	P.AAV, LOC_DESC+4		
				AE	9F	000DB	PUSHAB	LOC_DESC		
	0000V	CF		53	DD	000DE	PUSHL	R3		
		5B		02	FB	000E0	CALLS	#2, RMS_OUTPUT		
				50	D0	000E5	MOVL	R0, STATUS		
			00B9	31	000E8	BRW	28\$			
			OC	AE	B4	000EB	CLRW	LOC_DESC	3839	
			10	AE	D4	000EE	CLRL	LOC_DESC+4	3840	
				67	D5	000F1	TSTL	(R7)	3841	
				1A	12	000F3	BNEQ	20\$		
				55	D4	000F5	CLRL	I	3843	
				10	11	000F7	BRB	19\$		
			OC	AE	9F	000F9	PUSHAB	LOC_DESC		
				53	DD	000FC	PUSHL	R3		
	0000V	CF		02	FB	000FE	CALLS	#2, RMS_OUTPUT		
		5B		50	D0	00103	MOVL	R0, STATUS		
		52		56	C0	00106	ADDL2	DW2, CURR_PTR		
EC		55		58	F3	00109	AOBLEQ	LINES_TO_SKIP, I, 18\$		
				3F	11	0010D	BRB	24\$	3841	
00 B442				00	3B	0010F	SKPC	#0, DW2, @0(R4)[CURR_PTR]	3851	
	08	AE		50	D0	00115	MOVL	R0, REM_ATTR		
	04	AE		51	D0	00119	MOVL	R1, FND		
			08	AE	D5	0011D	TSTL	REM_ATTR	3853	
				14	13	00120	BEQL	21\$		
				52	DD	00122	PUSHL	CURR_PTR	3860	
			04	AE	9F	00124	PUSHAB	DW	3859	
				53	DD	00127	PUSHL	R3		
			10	AE	9F	00129	PUSHAB	FND		
			18	AE	9F	0012C	PUSHAB	REM_ATTR		
	0000V	CF		05	FB	0012F	CALLS	#5, DO_ATTR		
				18	11	00134	BRB	24\$	3853	
				55	D4	00136	CLRL	I	3863	
				10	11	00138	BRB	23\$		
			OC	AE	9F	0013A	PUSHAB	LOC_DESC		
				53	DD	0013D	PUSHL	R3		
	0000V	CF		02	FB	0013F	CALLS	#2, RMS_OUTPUT		
		5B		50	D0	00144	MOVL	R0, STATUS		
		52		56	C0	00147	ADDL2	DW2, CURR_PTR		
EC		55		58	F3	0014A	AOBLEQ	LINES_TO_SKIP, I, 22\$		
				52	DD	0014E	PUSHL	CURR_PTR	3870	
			04	AE	9F	00150	PUSHAB	DW		
	FC26	CF		02	FB	00153	CALLS	#2, TRIMMED_LENGTH		
		6E		50	D0	00158	MOVL	R0, DW		
				67	D5	0015B	TSTL	(R7)	3871	
				13	13	0015D	BEQL	25\$		
00 B442				00	3B	0015F	SKPC	#0, DW2, @0(R4)[CURR_PTR]	3879	
	08	AE		50	D0	00165	MOVL	R0, REM_ATTR		
	04	AE		51	D0	00169	MOVL	R1, FND		
			08	AE	D5	0016D	TSTL	REM_ATTR		
				17	12	00170	BNEQ	26\$		
	OC	AE		6E	B0	00172	MOVW	DW, LOC_DESC		
	10	AE		52	D0	00176	MOVL	CURR_PTR, LOC_DESC+4		
			OC	AE	9F	0017A	PUSHAB	LOC_DESC		
				53	DD	0017D	PUSHL	R3		
	0000V	CF		02	FB	0017F	CALLS	#2, RMS_OUTPUT		

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
PUT_MAP - Output current map

C 7
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 102
(29)

LI
V0

5B	50	D0	00184	MOVL	R0, STATUS
	12	11	00187	BRB	27\$
	52	DD	00189	PUSHL	CURR_PTR
	04	AE	9F 0018B	PUSHAB	DW
	53	DD	0018E	PUSHL	R3
	10	AE	9F 00190	PUSHAB	FND
	18	AE	9F 00193	PUSHAB	REM_ATTR
0000V	CF	05	FB 00196	CALLS	#5, DO_ATTR
	52	56	C0 0019B	ADDL2	DW2, CORR_PTR
	6E	56	D0 0019E	MOVL	DW2, DW
		FE89	31 001A1	BRW	1\$
	67 00000000	EF	D0 001A4	MOVL	SCR\$L_WORKMASK, (R7)
	50	5B	D0 001AB	MOVL	STATUS, R0
		04	001AE	RET	

3885
3886
3754
3889
3890
3891

; Routine Size: 431 bytes, Routine Base: _LIB\$CODE + 0C65

; 3662 3892 1 !<BLF/PAGE>


```
3664 3893 1 %SBTTL 'DO_ATTR - Handle bold and underlining'
3665 3894 1 ROUTINE DO_ATTR (
3666 3895 1     BYTES_REMAINING,
3667 3896 1     FNZ,
3668 3897 1     TCB: REF BLOCK [, BYTE],
3669 3898 1     LINE_LENGTH,
3670 3899 1     LINE_ADDR
3671 3900 1 ) =
3672 3901 1
3673 3902 1 ++
3674 3903 1 FUNCTIONAL DESCRIPTION:
3675 3904 1     This routine handles lines containing underlining or bolding.
3676 3905 1     A larger line is built containing concatenated copies of the
3677 3906 1     original line with embedded <CR>'s. Bold characters are
3678 3907 1     overprinted, underlined characters are converted to '-' and
3679 3908 1     then overprinted.
3680 3909 1
3681 3910 1 CALLING SEQUENCE:
3682 3911 1
3683 3912 1     ret_status.wlc.v = DO_ATTR (BYTES_REMAINING.rl.r,
3684 3913 1                               FNZ.rl.r,
3685 3914 1                               TCB.rab.r,
3686 3915 1                               LINE_LENGTH.rl.r
3687 3916 1                               LINE_ADDR.ra.r)
3688 3917 1
3689 3918 1 FORMAL PARAMETERS:
3690 3919 1
3691 3920 1     BYTES_REMAINING.rl.r    Count of number of bytes remaining in
3692 3921 1                             current line. It is assumed to be less
3693 3922 1                             than or equal to DEV_WIDTH.
3694 3923 1
3695 3924 1     FNZ.rl.r               Address of first non_zero attribute byte
3696 3925 1                             in a table of attribute bytes.
3697 3926 1
3698 3927 1     TCB.rab.r             The current terminal control block.
3699 3928 1                             Used to calculate the size of the
3700 3929 1                             mapped data area and hence the start of
3701 3930 1                             the attribute area.
3702 3931 1
3703 3932 1     LINE_LENGTH.rl.r      Length of current line
3704 3933 1
3705 3934 1     LINE_ADDR.ra.r        Address of current line
3706 3935 1
3707 3936 1
3708 3937 1 IMPLICIT INPUTS:
3709 3938 1
3710 3939 1     NONE
3711 3940 1
3712 3941 1 IMPLICIT OUTPUTS:
3713 3942 1
3714 3943 1     NONE
3715 3944 1
3716 3945 1 COMPLETION STATUS:
3717 3946 1
3718 3947 1     $$$ NORMAL           Normal successful completion
3719 3948 1     Errors returned by:  RMS_OUTPUT
3720 3949 1                          STR$DUPL_CHAR
```

```
3721 3950 1 |
3722 3951 1 | SIDE EFFECTS:
3723 3952 1 |
3724 3953 1 | NONE
3725 3954 1 | --
3726 3955 1 |
3727 3956 2 | BEGIN
3728 3957 2 | BUILTIN
3729 3958 2 | SKPC;
3730 3959 2 | LOCAL
3731 3960 2 | BR,
3732 3961 2 | LOC FNZ : REF BLOCK [, BYTE],
3733 3962 2 | STATUS,
3734 3963 2 | UNDER_INDEX,
3735 3964 2 |
3736 3965 2 |
3737 3966 2 | BOLD_INDEX,
3738 3967 2 |
3739 3968 2 | UNDER_DESC : BLOCK [8, BYTE],
3740 3969 2 |
3741 3970 2 | BOLD_DESC : BLOCK [8, BYTE],
3742 3971 2 |
3743 3972 2 | FIX_UNDER_DESC : BLOCK [8, BYTE],
3744 3973 2 |
3745 3974 2 |
3746 3975 2 | FIX_BOLD_DESC : BLOCK [8, BYTE],
3747 3976 2 |
3748 3977 2 |
3749 3978 2 | CR_DESC : BLOCK [8, BYTE],
3750 3979 2 |
3751 3980 2 | RESULT_DESC : BLOCK [8, BYTE];
3752 3981 2 |
3753 3982 2 | +
3754 3983 2 | Construct descriptor for and allocate space for the underline buffer.
3755 3984 2 | Initialize buffer to all blanks. Initialize fixed underline buffer
3756 3985 2 | descriptor except for length which we compute as we procede.
3757 3986 2 | -
3758 3987 2 | UNDER_DESC [DSC$B_CLASS] = DSC$K_CLASS_D ; ! Dynamic
3759 3988 2 | UNDER_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ; ! Text
3760 3989 2 | UNDER_DESC [DSC$W_LENGTH] = 0 ; ! Length
3761 3990 2 | UNDER_DESC [DSC$A_POINTER] = 0 ; ! Unallocated address
3762 3991 2 |
3763 3992 2 | STATUS = STR$DUPL_CHAR ( UNDER_DESC, ! Resulting descriptor
3764 3993 2 | .LINE_LENGTH, ! Length needed
3765 3994 2 | UPLIT (BYTE (' '))); ! Fill character
3766 3995 2 |
3767 3996 2 | FIX_UNDER_DESC [DSC$B_CLASS] = DSC$K_CLASS_S ; ! Fixed
3768 3997 2 | FIX_UNDER_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ; ! Text
3769 3998 2 | FIX_UNDER_DESC [DSC$A_POINTER] = .UNDER_DESC [DSC$A_POINTER] ;
3770 3999 2 |
3771 4000 2 |
3772 4001 2 | +
3773 4002 2 | Construct descriptor for and allocate space for the bolding buffer.
3774 4003 2 | Initialize buffer to all blanks. Initialize fixed bolding buffer
3775 4004 2 | descriptor except for length which we compute as we procede.
3776 4005 2 | -
3777 4006 2 | BOLD_DESC [DSC$B_CLASS] = DSC$K_CLASS_D ; ! Dynamic
```

```

: 3778      4007 2      BOLD_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;      ! Text
: 3779      4008 2      BOLD_DESC [DSC$W_LENGTH] = 0 ;                ! Length
: 3780      4009 2      BOLD_DESC [DSC$A_POINTER] = 0 ;                ! Unallocated address
: 3781      4010 2
: 3782      4011 2      STATUS = STR$DUPL_CHAR ( BOLD_DESC,            ! Resulting descriptor
: 3783      4012 2              .LINE_LENGTH,                        ! Length needed
: 3784      4013 2              UPLIT-(BYTE (' ')));                ! Fill character
: 3785      4014 2
: 3786      4015 2      FIX_BOLD_DESC [DSC$B_CLASS] = DSC$K_CLASS_S ;      ! Fixed
: 3787      4016 2      FIX_BOLD_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;      ! Text
: 3788      4017 2      FIX_BOLD_DESC [DSC$A_POINTER] = .BOLD_DESC [DSC$A_POINTER] ;
: 3789      4018 2
: 3790      4019 2
: 3791      4020 2      + Initialize indices into BUFFER. These are the relative positions of
: 3792      4021 2      the right-most underline char, and right-most 'char-for-bolding'.
: 3793      4022 2      They may get reset as we proceed with loop below.
: 3794      4023 2      -
: 3795      4024 2      BOLD_INDEX = -1;
: 3796      4025 2      UNDER_INDEX = -1;
: 3797      4026 2
: 3798      4027 2      +
: 3799      4028 2      Loop through all the attributes bytes remaining, building carriage
: 3800      4029 2      returns and properly positioned copies of the original text that
: 3801      4030 2      needs to be bolded. For underlining, we build a copy of the original
: 3802      4031 2      text that contains blanks in all positions except those to be
: 3803      4032 2      underlined. These cells will contain an underscore.
: 3804      4033 2      We manually break out of the loop when there are no more non-zero
: 3805      4034 2      attribute bytes to process.
: 3806      4035 2      -
: 3807      4036 2      LOC_FNZ = .FNZ ;
: 3808      4037 2      BR = .BYTES_REMAINING ;      ! Initialize local copy
: 3809      4038 2      WHILE -1
: 3810      4039 2      DO
: 3811      4040 3      BEGIN      ! Overall loop
: 3812      4041 3      LOCAL
: 3813      4042 3      INDEX;      ! Local index into underline buffer and
: 3814      4043 3      ! bold buffer
: 3815      4044 3      INDEX = .TCB [SCR$W_DEVWIDTH] - .BR ;
: 3816      4045 3      IF .LOC_FNZ [SCR$V_UNDERLINE] EQL 1
: 3817      4046 3      THEN
: 3818      4047 4      BEGIN      ! Underlining needed
: 3819      4048 4      UNDER_INDEX = .INDEX ;      ! Right-most underline
: 3820      4049 4      .UNDER_DESC [DSC$A_POINTER] + .INDEX = %C'_ ' ;
: 3821      4050 3      END;      ! Underlining needed
: 3822      4051 3
: 3823      4052 3      IF .LOC_FNZ [SCR$V_BOLD] EQL 1
: 3824      4053 3      THEN
: 3825      4054 4      BEGIN      ! Bolding needed
: 3826      4055 4      .BOLD_DESC [DSC$A_POINTER] + .INDEX =
: 3827      4056 4      .(.LINE_ADDR + .INDEX) ;
: 3828      4057 4      BOLD_INDEX = .INDEX ;      ! Right-most bold
: 3829      4058 3      END;      ! Bolding needed
: 3830      4059 3
: 3831      4060 3      BR = .BR - 1 ;
: 3832      4061 3      LOC_FNZ = .LOC_FNZ + 1 ;      ! to next attribute byte
: 3833      4062 3      SKPT (%REF(NULL), BR, .LOC_FNZ; BR, LOC_FNZ);
: 3834      4063 3      IF .BR EQL 0
```

```

3835 4064 3      THEN
3836 4065 3          EXITLOOP ;
3837 4066 2      END;      . Overall loop
3838 4067 2
3839 4068 2      +
3840 4069 2      Calculate true lengths of underline and bold buffers.
3841 4070 2      -
3842 4071 2      FIX_UNDER_DESC [DSC$W_LENGTH] = .UNDER_INDEX + 1 ;
3843 4072 2      FIX_BOLD_DESC [DSC$W_LENGTH] = .BOLD_INDEX + 1 ;
3844 4073 2
3845 4074 2      +
3846 4075 2      Build composite string by concatenating the pieces.
3847 4076 2      -
3848 4077 2      RESULT_DESC [DSC$B_CLASS] = DSC$K_CLASS_D ;      ! Dynamic
3849 4078 2      RESULT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;      ! Text
3850 4079 2      RESULT_DESC [DSC$W_LENGTH] = 0 ;                  ! Length
3851 4080 2      RESULT_DESC [DSC$A_POINTER] = 0 ;                ! Unallocated address
3852 4081 2
3853 4082 2      CR_DESC [DSC$B_CLASS] = DSC$K_CLASS_S ;          ! Fixed
3854 4083 2      CR_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;          ! Text
3855 4084 2      CR_DESC [DSC$W_LENGTH] = 1 ;                      ! Length = 1
3856 4085 2      CR_DESC [DSC$A_POINTER] = UPLIT (BYTE (CR));      ! A <CR>
3857 4086 2
3858 4087 3      BEGIN
3859 4088 3      LOCAL
3860 4089 3          DESC : BLOCK [8,BYTE];
3861 4090 3      DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
3862 4091 3      DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
3863 4092 3      DESC [DSC$W_LENGTH] = ..LINE_LENGTH;
3864 4093 3      DESC [DSC$A_POINTER] = .LINE_ADDR;
3865 4094 3
3866 4095 3      STATUS = STR$CONCAT ( RESULT_DESC,      ! Resulting string
3867 4096 3          DESC,      ! Original string
3868 4097 3          CR_DESC,      ! A <CR>
3869 4098 3          FIX_UNDER_DESC,      ! Underlining string
3870 4099 3          CR_DESC,      ! A <CR>
3871 4100 3          FIX_BOLD_DESC,      ! Bolding string copy 1
3872 4101 3          CR_DESC,      ! A <CR>
3873 4102 3          FIX_BOLD_DESC,      ! Bolding string copy 2
3874 4103 3          CR_DESC,      ! A <CR>
3875 4104 3          FIX_BOLD_DESC ) ;      ! Bolding string copy 3
3876 4105 3
3877 4106 2      END;
3878 4107 2
3879 4108 2      +
3880 4109 2      Output composite string we've built
3881 4110 2      -
3882 4111 2      STATUS = RMS_OUTPUT ( .TCB, RESULT_DESC );
3883 4112 2
3884 4113 2      +
3885 4114 2      Give back strings used
3886 4115 2      -
3887 4116 2      LIB$FREE1_DD ( UNDER_DESC ) ;
3888 4117 2      LIB$FREE1_DD ( BOLD_DESC ) ;
3889 4118 2      LIB$FREE1_DD ( RESULT_DESC ) ;
3890 4119 2      RETURN (.STATUS) ;
3891 4120 1      END;      ! End of routine DO_ATTR
```

:

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

			0C	AE	D4	000B7	CLRL	RESULT_DESC+4	:	4080
10	AE	010E	0001	8F	D0	000BA	MOVL	#17694721, CR_DESC	:	4084
14	AE		FF39	CF	9E	000C2	MOVAB	P.AAY, CR_DESC+4	:	4085
02	AE		010E	8F	B0	000C8	MOVW	#270, DESC+2	:	4090
	6E		10	BC	R0	000CE	MOVW	@LINE_LENGTH, DESC	:	4092
04	AE		14	AC	D0	000D2	MOVL	LINE_ADDR, DESC+4	:	4093
			18	AE	9F	000D7	PUSHAB	FIX_BOLD_DESC	:	4095
			14	AE	9F	000DA	PUSHAB	CR_DESC	:	
			20	AE	9F	000DD	PUSHAB	FIX_BOLD_DESC	:	
			1C	AE	9F	000E0	PUSHAB	CR_DESC	:	
			28	AE	9F	000E3	PUSHAB	FIX_BOLD_DESC	:	
			24	AE	9F	000E6	PUSHAB	CR_DESC	:	
			38	AE	9F	000E9	PUSHAB	FIX_UNDER_DESC	:	
			2C	AE	9F	000EC	PUSHAB	CR_DESC	:	
			20	AE	9F	000EF	PUSHAB	DESC	:	
			2C	AE	9F	000F2	PUSHAB	RESULT_DESC	:	
00000000G	0G			0A	FB	000F5	CALLS	#10, STR\$CONCAT	:	
	58			50	D0	000FC	MOVL	R0, STATUS	:	
			08	AE	9F	000FF	PUSHAB	RESULT_DESC	:	4110
			0C	AC	DD	00102	PUSHL	TCB	:	
0000V	CF			02	FB	00105	CALLS	#2, RMS_OUTPUT	:	
	58			50	D0	0010A	MOVL	R0, STATUS	:	
			30	AE	9F	0010D	PUSHAB	UNDER_DESC	:	4116
	69			01	FB	00110	CALLS	#1, LIB\$FREE1_DD	:	
			28	AE	9F	00113	PUSHAB	BOLD_DESC	:	4117
	69			01	FB	00116	CALLS	#1, LIB\$FREE1_DD	:	
			08	AE	9F	00119	PUSHAB	RESULT_DESC	:	4118
	69			01	FB	0011C	CALLS	#1, LIB\$FREE1_DD	:	
	50			58	D0	0011F	MOVL	STATUS, R0	:	4119
				04	00	122	RET		:	4120

; Routine Size: 291 bytes, Routine Base: _LIB\$CODE + 0E1D

; 3892 4121 1 !<BLF/PAGE>

```
3894 4122 1 $SBTTL 'RMS_OUTPUT - Write string via RMS'
3895 4123 1 ROUTINE RMS_OUTPUT ( TCB : REF BLOCK [,BYTE],
3896 4124 1     STRING : REF BLOCK [,BYTE] ) =
3897 4125 1 ++
3898 4126 1 FUNCTIONAL DESCRIPTION:
3899 4127 1
3900 4128 1     This routine writes the specified string to the terminal using
3901 4129 1     RMS. If an RMS file is open for the stream it will be used,
3902 4130 1     else LIB$PUT_OUTPUT will be called.
3903 4131 1
3904 4132 1 CALLING SEQUENCE:
3905 4133 1
3906 4134 1     ret_status.wlc.v = RMS_OUTPUT ( TCB.rab.r, string.rt.ds )
3907 4135 1
3908 4136 1 FORMAL PARAMETERS:
3909 4137 1
3910 4138 1     TCB.rab.r      Address of current terminal control block.
3911 4139 1
3912 4140 1     STRING.rt.ds   Address of fixed string descriptor of output
3913 4141 1     string.
3914 4142 1
3915 4143 1 IMPLICIT INPUTS:
3916 4144 1
3917 4145 1     NONE
3918 4146 1
3919 4147 1 IMPLICIT OUTPUTS:
3920 4148 1
3921 4149 1     NONE
3922 4150 1
3923 4151 1 ROUTINE STATUS:
3924 4152 1
3925 4153 1     Status returned by RMS PUT, LIB$PUT_OUTPUT, or users output
3926 4154 1     routine.
3927 4155 1
3928 4156 1 SIDE EFFECTS:
3929 4157 1
3930 4158 1     String will be output.
3931 4159 1 --
3932 4160 1
3933 4161 2 BEGIN
3934 4162 2
3935 4163 2 BUILTIN
3936 4164 2 CALLG;
3937 4165 2
3938 4166 2 LOCAL
3939 4167 2 STATUS;                ! Status to return to caller
3940 4168 2
3941 4169 2 IF .TCB [SCR$L_RTNADDR] EQL 0    ! If no user-specified call-back
3942 4170 2 THEN
3943 4171 3 BEGIN ! We do ouptut
3944 4172 3 LOCAL
3945 4173 3 RAB : REF $RAB_DECL;
3946 4174 3
3947 4175 3 IF .TCB [SCR$L_RAB] EQL 0      ! If file not open on channel
3948 4176 3 THEN
3949 4177 4 BEGIN ! Use LIB$PUT_OUTPUT instead
3950 4178 4 STATUS = LIB$PUT_OUTPUT ( .STRING ) ;
```

```
3951 4179 4      END      ! Use LIB$PUT_OUTPUT instead
3952 4180 3      ELSE
3953 4181 4          BEGIN      ! Do via RMS
3954 4182 4              RAB = .TCB [SCR$L_RAB];
3955 4183 4              RAB [RAB$W_RSZ] = .STRING [DSC$W_LENGTH] ; ! length of string
3956 4184 4              RAB [RAB$L_RBF] = .STRING [DSC$A_POINTER] ; ! Addr of string
3957 4185 4
3958 4186 4              WHILE $PUT (RAB = .RAB) EQL RMSS_RSA DO $WAIT (RAB = .RAB) ;
3959 4187 4              STATUS = .RAB [RAB$L_STS] ;
3960 4188 3          END      ! Do via RMS
3961 4189 3      END      ! We do output
3962 4190 3
3963 4191 2      ELSE
3964 4192 2          BEGIN      ! User does output
3965 4193 2              LOCAL
3966 4194 2                  CHANNEL,      ! Channel as a longword
3967 4195 2                  PARM_LIST : VECTOR [5];      ! Parameter list for call-back routine
3968 4196 2
3969 4197 2              (CHANNEL = .TCB [SCR$W_CHAN] ;      ! Extract channel as longword
3970 4198 2
3971 4199 2              !+
3972 4200 2              ! Bliss must be forced to generate the proper CALL with R0 (not
3973 4201 2              ! the contents of R0) being moved to STATUS.
3974 4202 2              !-
3975 4203 2              PARM_LIST [0] = 4;      ! number of args
3976 4204 2              PARM_LIST [1] = .TCB [SCR$L_RTNARG];      ! user-supplied arg
3977 4205 2              PARM_LIST [2] = CHANNEL;      ! addr of channel
3978 4206 2              PARM_LIST [3] = .STRING;      ! addr of output string
3979 4207 2              PARM_LIST [4] = TCB [SCR$L_STREAM];      ! addr of stream
3980 4208 2              STATUS = CALLG (PARM_LIST, .TCB [SCR$L_RTNADDR]);      ! call user routine
3981 4209 2
3982 4210 2          END;      ! User does output
3983 4211 2
3984 4212 2      RETURN (.STATUS) ;
3985 4213 2
3986 4214 2
3987 4215 1      END;      ! End of routine RMS_OUTPUT
```

.EXTRN SYSS\$PUT, SYSS\$WAIT

```
003C 00000 RMS_OUTPUT:
      5E      18      C2 00002      .WORD      Save R2,R3,R4,R5      4123
      54      08      AC D0 00005      SUBL2      #24, SP
      52      04      AC D0 00009      MOVL      STRING, R4      4178
      38      A2      D5 0000D      MOVL      TCB, R2      4169
      40      12      00010      TSTL      56(R2)
      74      A2      D5 00012      BNEQ      4$
      08      12      00015      TSTL      116(R2)      4175
      54      DD      00017      BNEQ      1$
      01      FB      00019      PUSHL      R4      4178
      4F      11      00020      CALLS      #1, LIB$PUT_OUTPUT
      74      A2      D0 00022 1$:      BRB      5$
      22      53      74      A2      D0 00022      MOVL      116(R2), RAB      4182
      28      A3      64      B0 00026      MOVW      (R4), 34(RAB)      4183
      04      A4      D0 0002A      MOVL      4(R4), 40(RAB)      4184
```


LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
RMS_OUTPUT - Write string via RMS

L 7
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 111
(31)

00000000G	00		53	DD	0002F	2\$:	PUSHL	RAB		4186
000182DA	8F		01	FB	00031		CALLS	#1, SYSSPUT		
			50	D1	00038		CMPL	R0, #99034		
			08	12	0003F		BNEQ	3\$		
00000000G	00		53	DD	00041		PUSHL	RAB		
			01	FB	00043		CALLS	#1, SYSSWAIT		
	55	08	E3	11	0004A		BRB	2\$		
			A3	D0	0004C	3\$:	MOVL	8(RAB), STATUS		4187
			22	11	00050		BRB	6\$		4169
	6E	08	A2	3C	00052	4\$:	MOVZWL	8(R2), CHANNEL		4198
04	AE		04	D0	00056		MOVL	#4, PARM_LIST		4204
08	AE	3C	A2	D0	0005A		MOVL	60(R2), PARM_LIST+4		4205
0C	AE		6E	9E	0005F		MOVAB	CHANNEL, PARM_LIST+8		4206
10	AE		54	D0	00063		MOVL	R4, PARM_LIST+12		4207
14	AE	30	A2	9E	00067		MOVAB	48(R2), PARM_LIST+16		4208
38	B2	04	AE	FA	0006C		CALLG	PARM_LIST, @56(R2)		4209
	55		50	D0	00071	5\$:	MOVL	R0, STATUS		
	50		55	D0	00074	6\$:	MOVL	STATUS, R0		4213
			04	00077			RET			4215

; Routine Size: 120 bytes, Routine Base: _LIB\$CODE + 0F40

; 3988 4216 1 !<BLF/PAGE>

```
3990 4217 1 %SBTTL 'SET_CURSOR - Create set cursor sequence'
3991 4218 1 ROUTINE SET_CURSOR (
3992 4219 1     TCB : REF BLOCK [,BYTE],
3993 4220 1     LINE_NO,
3994 4221 1     COL_NO,
3995 4222 1     BUFFER,
3996 4223 1     CUR_SIZE
3997 4224 1 ) =
3998 4225 1
3999 4226 1 ++
4000 4227 1 FUNCTIONAL DESCRIPTION:
4001 4228 1     This routine generates the escape sequence for a set cursor
4002 4229 1     position and appends the string to a given output buffer. No
4003 4230 1     buffer overflow checking is done as it is assumed that the cursor
4004 4231 1     will be the first thing in the buffer.
4005 4232 1
4006 4233 1 CALLING SEQUENCE:
4007 4234 1
4008 4235 1     ret_status.wlc.v = SET_CURSOR (TCB.rz.r, LINE_NO.rl.v, COL_NO.rl.v,
4009 4236 1     BUFFER.mt.r, CUR_SIZE.ml.r)
4010 4237 1
4011 4238 1 FORMAL PARAMETERS:
4012 4239 1
4013 4240 1     TCB.rz.r          addr of Terminal Control Block
4014 4241 1     LINE_NO.rl.v     line number
4015 4242 1     COL_NO.rl.v      column number
4016 4243 1     BUFFER.mt.r      addr of buffer
4017 4244 1     CUR_SIZE.ml.r    # bytes currently in buffer
4018 4245 1
4019 4246 1 IMPLICIT INPUTS:
4020 4247 1
4021 4248 1     NONE
4022 4249 1
4023 4250 1 IMPLICIT OUTPUTS:
4024 4251 1
4025 4252 1     NONE
4026 4253 1
4027 4254 1 COMPLETION STATUS:
4028 4255 1
4029 4256 1
4030 4257 1 SIDE EFFECTS:
4031 4258 1
4032 4259 1     NONE
4033 4260 1 --
4034 4261 1
4035 4262 2 BEGIN
4036 4263 2
4037 4264 2 LOCAL
4038 4265 2     VT100CTL : VECTOR [1, 8] INITIAL (
4039 4266 2         DSC$K_DTYPE T ^24 + DSC$K_CLASS S ^16 + 10,
4040 4267 2         UPLIT ( BYTE (ESC, LB, '!OL'; 'UL', VT100_SC))) ,
4041 4268 2         dsc for cvt to vt100 sequence
4042 4269 2         FAO control string
4043 4270 2     FREE_ADDR : REF VECTOR [,BYTE], addr of 1st free byte
4044 4271 2     TERM_TYPE; terminal type
4045 4272 2
4046 4273 2
```

```
4047 4274 2 FREE_ADDR = .BUFFER + ..CUR_SIZE; ! addr of next free byte
4048 4275 2
4049 4276 2 IF .TCB [SCR$B_TYPE] EQL VTFOREIGN
4050 4277 2 THEN
4051 4278 2 TERM_TYPE = .TCB [SCR$B_DEVTYP]
4052 4279 2 ELSE
4053 4280 2 TERM_TYPE = .TCB [SCR$B_TYPE];
4054 4281 2
4055 4282 2 CASE .TERM_TYPE FROM UNKNOWN TO VT100 OF
4056 4283 2 SET
4057 4284 2
4058 4285 2 [UNKNOWN]:
4059 4286 2 RETURN 1; ! do nothing
4060 4287 2
4061 4288 2 [VT05]:
4062 4289 2 BEGIN
4063 4290 2 .CUR_SIZE = ..CUR_SIZE + 7; ! update current size of buffer
4064 4291 2 FREE_ADDR [0] = VT05_SC; ! put set cursor sequence into buffer
4065 4292 2 FREE_ADDR [1] = CB + ..LINE_NO;
4066 4293 2 FREE_ADDR [2] = 0;
4067 4294 2 FREE_ADDR [3] = 0;
4068 4295 2 FREE_ADDR [4] = 0;
4069 4296 2 FREE_ADDR [5] = 0;
4070 4297 2 FREE_ADDR [6] = CB + ..COL_NO;
4071 4298 2 END;
4072 4299 2
4073 4300 2 [VT52]:
4074 4301 2 BEGIN
4075 4302 2 .CUR_SIZE = ..CUR_SIZE + 4; ! update current size of buffer
4076 4303 2 FREE_ADDR [0] = ESC; ! put set cursor sequence into buffer
4077 4304 2 FREE_ADDR [1] = VT52_SC;
4078 4305 2 FREE_ADDR [2] = CB + ..LINE_NO;
4079 4306 2 FREE_ADDR [3] = CB + ..COL_NO;
4080 4307 2 END;
4081 4308 2
4082 4309 2 [VT100]:
4083 4310 2 BEGIN
4084 4311 2 LOCAL
4085 4312 2 STATUS,
4086 4313 2 CVT_ARGS : VECTOR [2],
4087 4314 2 FAO_BUFFER : BLOCK [8, BYTE],
4088 4315 2 FAO_LEN : WORD;
4089 4316 2
4090 4317 2 CVT_ARGS [0] = ..LINE_NO;
4091 4318 2 CVT_ARGS [1] = ..COL_NO;
4092 4319 2 FAO_BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
4093 4320 2 FAO_BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
4094 4321 2 FAO_BUFFER [DSC$W_LENGTH] = (.TCB [SCR$L_BUFSIZ] - ..CUR_SIZE);
4095 4322 2 FAO_BUFFER [DSC$A_POINTER] = .FREE_ADDR;
4096 4323 2
4097 4324 2 !
4098 4325 2 ! Convert to ASCII characters and move to buffer.
4099 4326 2 !
4100 4327 2 STATUS = $FAOL (CTRSTR = VT100CTL, OUTLEN = FAO_LEN,
P 4101 4328 2 OUTBUF = FAO_BUFFER, PRMLST = CVT_ARGS);
4102 4329 2 IF NOT .STATUS THEN RETURN (.STATUS);
4103 4330 2 .CUR_SIZE = ..CUR_SIZE + .FAO_LEN; ! add length of appended string
```

```

: 4104      4331 3
: 4105      4332 2      END;
: 4106      4333 2
: 4107      4334 2      [INRANGE,OUTRANGE]:
: 4108      4335 2      RETURN 0;
: 4109      4336 2      ! should never get here
: 4110      4337 2
: 4111      4338 2      TES;
: 4112      4339 2      RETURN 1;
: 4113      4340 2
: 4114      4341 1      END;
                                ! End of routine SET_CURSOR
```

```

4C 55 21 3B 4C 5B 1B 00FB8 P.AAZ: .BYTE 27, 91
                    21 00FBA .ASCII \!UL;!UL\
                    66 00FC1 .BYTE 102
```

```

                                0004 00000 SET_CURSOR:
                                .WORD      Save R2
                                SUBL2      #28, SP
                                MOVL       #234946570, VT100CTL
                                MOVAB      P.AAZ, VT100CTL+4
                                ADDL3      @CUR_SIZE, BUFFER, FREE_ADDR
                                MOVL       TCB, R1
                                CMPB      10(R1), #4
                                BNEQ       1$
                                MOVZBL     11(R1), TERM_TYPE
                                BRB        2$
                                MOVZBL     10(R1), TERM_TYPE
                                CASEL      TERM_TYPE, #0, #3
                                .WORD      7$-3$, -
                                4$-3$, -
                                5$-3$, -
                                6$-3$
                                BRB        8$
                                ADDL2      #7, @CUR_SIZE
                                MOVB       #14, (FREE_ADDR)
                                ADDB3      #31, LINE_NO, 1(FREE_ADDR)
                                CLRL       2(FREE_ADDR)
                                ADDB3      #31, COL_NO, 6(FREE_ADDR)
                                BRB        7$
                                ADDL2      #4, @CUR_SIZE
                                MOVW       #22811, (FREE_ADDR)
                                ADDB3      #31, LINE_NO, 2(FREE_ADDR)
                                ADDB3      #31, COL_NO, 3(FREE_ADDR)
                                BRB        7$
                                MOVQ       LINE_NO, CVT_ARGS
                                MOVW       #270, FAO_BUFFER+2
                                SUBW3      @CUR_SIZE, 76(R1), FAO_BUFFER
                                MOVL       FREE_ADDR, FAO_BUFFER+4
                                PUSHAB     CVT_ARGS
                                PUSHAB     FAO_BUFFER
                                PUSHAB     FAO_LEN
                                PUSHAB     VT100CTL

                                66 11 00038      BRB
                                07 C0 0003A 4$:  ADDL2
                                0E 90 0003E      MOVB
                                1F 81 00041      ADDB3
                                A0 D4 00047      CLRL
                                1F 81 0004A      ADDB3
                                4A 11 00050      BRB
                                04 C0 00052 5$:  ADDL2
                                8F B0 00056      MOVW
                                1F 81 0005B      ADDB3
                                1F 81 00061      ADDB3
                                33 11 00067      BRB
                                AC 7D 00069 6$:  MOVQ
                                8F B0 0006E      MOVW
                                BC A3 00074      SUBW3
                                50 D0 0007B      MOVL
                                AC 9F 0007F      PUSHAB
                                08 AE 9F 00082      PUSHAB
                                08 AE 9F 00085      PUSHAB
                                20 AE 9F 00088      PUSHAB

                                14 BC          66 11 00038
                                60          07 C0 0003A 4$:
                                08 AC          0E 90 0003E
                                02          1F 81 00041
                                A0 D4 00047
                                1F 81 0004A
                                4A 11 00050
                                14 BC          04 C0 00052 5$:
                                60          8F B0 00056
                                08 AC          1F 81 0005B
                                0C AC          1F 81 00061
                                08 AE          33 11 00067
                                06 AE          AC 7D 00069 6$:
                                4C A1          8F B0 0006E
                                08 AE          BC A3 00074
                                14          50 D0 0007B
                                AC AE          AC 9F 0007F
                                08 AE          08 AE 9F 00082
                                08 AE          08 AE 9F 00085
                                20 AE          20 AE 9F 00088

                                50          14 AE 0E01000A
                                18 AE          E6 AF 9E 0000D
                                10 AC          14 BC C1 00012
                                51          04 AC D0 00018
                                04          0A A1 91 0001C
                                52          0B A1 9A 00022
                                52          0A A1 9A 00026
                                00          52 CF 0002C 1$:
                                0039 0022 000A 006C 00030 3$:
                                000A          00030 3$:

                                01 A0          14 BC
                                06 A0          60
                                02 A0          08 AC
                                03 A0          0C AC
                                04 AE          14 BC
                                08 AE          60
                                0C AE          08 AC
                                14 AE          0C AC
                                20 AE          08 AE
                                24 AE          06 AE
                                28 AE          4C A1
                                32 AE          08 AE
                                36 AE          14 AE
                                40 AE          0C AE
                                44 AE          08 AE
                                48 AE          20 AE
```

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SET_CURSOR - Create set cursor sequence

C 8
16-Sep-1984 02:28:18
14-Sep-1984 13:34:42

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]SCREEN.B32;1

Page 115
(32)

00000000G	00	04	FB	0008B	CALLS	#4, SYSS\$FAOL
	0D	50	E9	00092	BLBC	STATUS, 9\$
	50	6E	3C	00095	MOVZWL	FAO_LEN, R0
14	BC	50	C0	00098	ADDL2	R0, @CUR_SIZE
	50	01	D0	0009C	MOVL	#1, R0
			04	0009F	RET	
		50	D4	000A0	CLRL	R0
			04	000A2	RET	

: 4329
: 4330
: 4339
: 4341
:

; Routine Size: 163 bytes, Routine Base: _LIB\$CODE + 0FC2

; 4115 4342 1 !<BLF/PAGE>

LIB\$SCREEN
V04-000

LIB\$SCREEN - Screen Management
SET_CURSOR - Create set cursor sequence

D 8
16-Sep-1984 02:28:18 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:34:42 [VMSLIB.SRC]SCREEN.B32;1

Page 116
(33)

: 4117 4343 1 END
: 4118 4344 1
: 4119 4345 0 ELUDOM

. End of module LIB\$SCREEN

SCR\$ERASE== SCR\$ERASE PAGE
LIB\$UP_SCROLL== SCR\$UP_SCROLL
LIB\$SET_BUFFER== SCR\$SET_BUFFER
LIB\$DOWN_SCROLL== SCR\$DOWN_SCROLL

PSECT SUMMARY

Name	Bytes	Attributes
_LIB\$DATA	12	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_LIB\$CODE	4197	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	37	0	581	00:01.0
_ \$255\$DUA28:[VMSLIB.OBJ]SCRLIB.L32;1	62	49	79	7	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SCREEN/OBJ=OBJ\$:SCREEN MSRC\$:SCREEN/UPDATE=(ENH\$:SCREEN)

: Size: 4057 code + 152 data bytes
: Run Time: 01:24.8
: Elapsed Time: 01:29.4
: Lines/CPU Min: 3073
: Lexemes/CPU-Min: 19800
: Memory Used: 264 pages
: Compilation Complete

0436 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0437

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL
CONFIDE

EQUIPMENT
INITIAL AND

CORPORATION
PROPRIETARY

SHRMSG
LTS